



PATENT ABSTRACTS OF JAPAN

(11) Publication number: **2002157291 A**(43) Date of publication of application: **31.05.02**

(51) Int. Cl. **G06F 17/50**
H01L 21/82
H01L 27/04
H01L 21/822

(21) Application number: **2000349938**(22) Date of filing: **16.11.00**(71) Applicant: **MATSUSHITA ELECTRIC IND CO LTD**(72) Inventor: **TAKEMURA KAZUYOSHI**

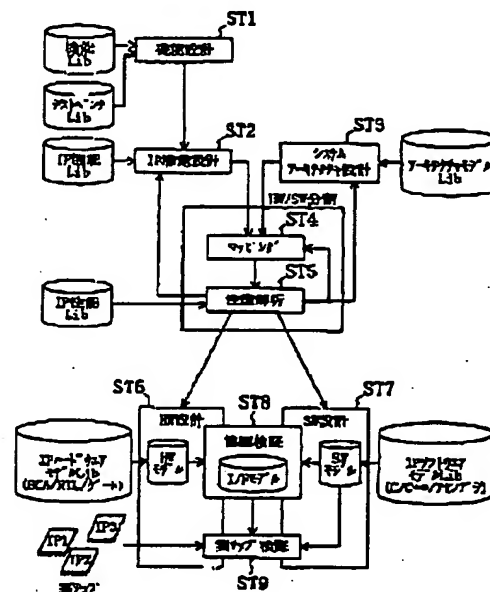
(54) **METHOD FOR DESIGNING SEMICONDUCTOR
 INTEGRATED CIRCUIT DEVICE**

(57) Abstract:

PROBLEM TO BE SOLVED: To provide a method for designing a semiconductor integrated circuit device for selecting IP flexibly while securing the function of a system surely.

SOLUTION: After performing function design and IP function design, the architecture design of the system is performed. After mapping for assigning respective functions (function blocks) to the component of architecture, performance analysis is performed. Next, HW design, SW design, HW/SW cooperative verification, real chip verification are performed. By performing hierarchical development of an IP function at a function block level in the IP function design in particular, HW/SW division by each function block becomes possible and the system is optimized. Furthermore, dynamic analysis with switching of HW/SW by each function block becomes possible.

COPYRIGHT: (C)2002,JPO



THIS PAGE BLANK (USPTO)

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-157291

(P2002-157291A)

(43) 公開日 平成14年5月31日 (2002.5.31)

(51) Int.Cl.	識別記号	F I	キーワード(参考)
G 0 6 F 17/50	6 5 4	G 0 6 F 17/50	6 5 4 M 5 B 0 4 6
			6 5 4 G 5 F 0 3 8
			6 5 4 K 5 F 0 6 4
	6 5 2		6 5 2 C
	6 5 8		6 5 8 T
審査請求 有 請求項の数18 O L (全 24 頁) 最終頁に続く			

(21) 出願番号 特願2000-349938(P2000-349938)

(22) 出願日 平成12年11月16日 (2000.11.16)

(71) 出願人 000005821

松下電器産業株式会社

大阪府門真市大字門真1006番地

(72) 発明者 竹村 和祥

大阪府門真市大字門真1006番地 松下電器
産業株式会社内

(74) 代理人 100077931

弁理士 前田 弘 (外7名)

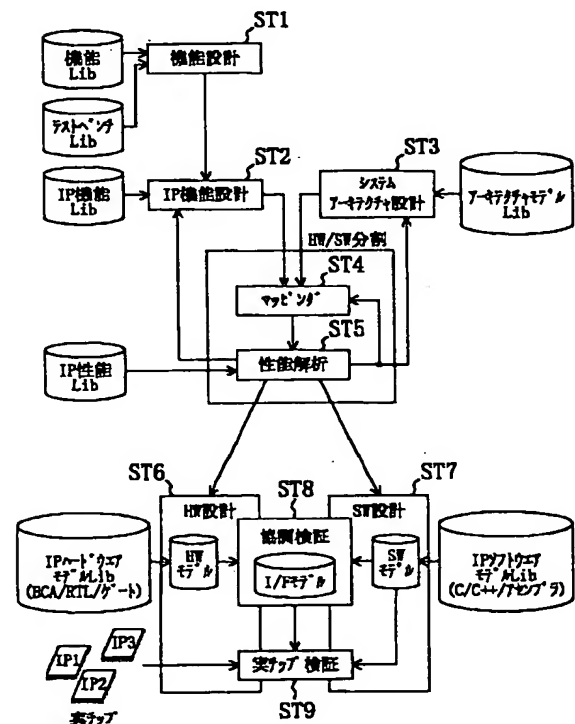
最終頁に続く

(54) 【発明の名称】 半導体集積回路装置の設計方法

(57) 【要約】

【課題】 システムの機能を確実に確保しながら I P の柔軟な選択を行なうための半導体集積回路装置の設計方法を提供する。

【解決手段】 機能設計と、I P 機能設計を行なったあと、システムのアーキテクチャ設計を行なう。各機能(機能ブロック)をアーキテクチャのコンポーネントに割り当てるマッピングを行なった後、性能解析を行なう。次に、HW設計と、SW設計と、HW/SW協調検証と、実チップ検証とを行なう。特に、I P 機能設計において I P 機能を機能ブロックレベルに階層展開を行なうことにより、機能ブロックごとに HW/SW 分割することが可能になり、より最適化されたシステムが得られる。また、機能ブロックごとに HW/SW の切り換えを伴う動的解析も可能になる。



【特許請求の範囲】

【請求項 1】 共通の機能を有する複数の IP を、複数の機能について記憶装置に格納するステップ (a) と、ある仕様を満たすための機能群構造を構築するステップ (b) と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を共通に有する複数の IP から 1 つの IP を選択して取り出して、上記機能群構造中の各機能と置き換えるステップ (c) とを含む半導体集積回路装置の設計方法。

【請求項 2】 請求項 1 記載の半導体集積回路装置の設計方法において、

上記各機能ごとに、当該機能の実現対象をデータベースのアーキテクチャモデル中の HW モデルと SW モデルとに割り当てるように分割するステップ (e) とをさらに含むことを特徴とする半導体集積回路装置の設計方法。

【請求項 3】 請求項 2 記載の半導体集積回路装置の設計方法において、

上記ステップ (c) の後、上記 IP の階層を機能ブロックレベルに展開するステップ (d) をさらに含み、

上記ステップ (e) では、上記展開された各機能ブロックごとに、当該機能ブロックの実現対象をアーキテクチャモデル中の HW モデルと SW モデルとに割り当てることを特徴とする半導体集積回路装置の設計方法。

【請求項 4】 各々ある機能を有する複数の IP を記憶装置に格納するステップ (a) と、

ある仕様を満たすための機能群構造を構築するステップ (b) と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を有する IP を選択して取り出して、上記機能群構造中の各機能と置き換えるステップ (c) と、

上記 IP を機能ブロックレベルに階層展開するステップ (d) と、

上記階層展開された各機能ブロックごとに、当該機能ブロックの実現化対象をデータベースのアーキテクチャモデル中の HW (ハードウェア) モデルと SW (ソフトウェア) モデルとに割り当てるように分割するステップ (e) とを含む半導体集積回路装置の設計方法。

【請求項 5】 請求項 4 記載の半導体集積回路装置の設計方法において、

上記ステップ (e) では、機能ブロックの HW の部分から面積の重複を考慮した静的解析を行なうことを特徴とする半導体集積回路装置の設計方法。

【請求項 6】 請求項 4 記載の半導体集積回路装置の設計方法において、

上記ステップ (e) の後、割り当てられた上記機能群構造中の各機能ブロックの HW モデルと SW モデルとへの割り当てを決定するための動的性能解析を行なうステップ (f) をさらに含むことを特徴とする半導体集積回路装置の設計方法。

【請求項 7】 請求項 6 記載の半導体集積回路装置の設計方法において、

上記データベースには、HW の機能毎に当該 HW と等価な機能を有する SW モデルを格納しておくことを特徴とする半導体集積回路装置の設計方法。

【請求項 8】 請求項 7 記載の半導体集積回路装置の設計方法において、

上記ステップ (f) では、時間軸に沿って消費電力が常に上限値を超えないように各機能ブロックの HW モデルと SW モデルとへの割り当てを切り換えることを特徴とする半導体集積回路装置の設計方法。

【請求項 9】 請求項 5 記載の半導体集積回路装置の設計方法において、

上記ステップ (f) では、ある機能ブロックを HW に割り当てたときの CPU の負荷率がある下限値よりも小さいときは、当該機能ブロックの割り当てを SW に切り換えることを特徴とする半導体集積回路装置の設計方法。

【請求項 10】 請求項 5～9 のうちいずれか 1 つに記載の半導体集積回路装置の設計方法において、

上記ステップ (e) では、メモリの消費電力を考慮した解析を行なうことを特徴とする半導体集積回路装置の設計方法。

【請求項 11】 複数の HW のコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ (a) と、

上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ (b) とを含み、

上記ステップ (b) では、コンポーネント群に複数のバスと各バス間を接続するバスブリッジモデルとが含まれるように上記アーキテクチャを構築することを特徴とする半導体集積回路装置の設計方法。

【請求項 12】 請求項 11 記載の半導体集積回路装置の設計方法において、

上記ステップ (b) では、上記バスブリッジモデルが、バス幅が互いに異なる 2 つのバス間をバス幅及びデータ転送速度を調整するように接続していることを特徴とする半導体集積回路装置の設計方法。

【請求項 13】 複数の HW のコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ (a) と、

上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ (b) と、

上記仕様を満たすための機能群構造を構築するステップ (c) と、

上記機能群構造の入力側及び出力側にテストベンチを付設するステップ (d) と、

上記機能群構造中の機能及びテストベンチを上記アーキテクチャ中のコンポーネントに割り当てるステップ (e) とを含む半導体集積回路装置の設計方法。

【請求項14】 請求項13記載の半導体集積回路装置の設計方法において、

上記ステップ (a) では、上記複数のコンポーネントの中にI/Fモデルを含めておき、

上記ステップ (b) では、上記コンポーネント群にI/Fモデルを含めるように上記アーキテクチャを構築することを特徴とする半導体集積回路装置の設計方法。

【請求項15】 請求項14記載の半導体集積回路装置の設計方法において、

上記ステップ (e) では、上記入力側又は出力側のテストベンチを上記アーキテクチャ中のI/Fモデルに割り当てることを特徴とする半導体集積回路装置の設計方法。

【請求項16】 請求項13～15のうちいずれか1つに記載の半導体集積回路装置の設計方法において、

上記ステップ (a) では、上記複数のコンポーネントの中にメモリを含めておき、

上記ステップ (b) では、上記コンポーネント群にメモリを含めるように上記アーキテクチャを構築し、

上記ステップ (e) では、上記入力側又は出力側のテストベンチを上記アーキテクチャ中のメモリに割り当てることを特徴とする半導体集積回路装置の設計方法。

【請求項17】 各々ある機能を有する複数のIPを記憶装置に格納するステップ (a) と、

ある仕様を満たすための機能群構造を構築するステップ (b) と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ (c) と、

上記IPを機能ブロックレベルに階層展開するステップ (d) と、

上記階層展開された各機能ブロックをアーキテクチャモデル中のコンポーネントに割り当てるステップ (e) と、

上記機能ブロックから当該機能ブロックの出力を受ける他の機能ブロックまでの経路を、上記アーキテクチャモデル中のコンポーネント間の経路として記述する機能ブロック間テーブルを作成するステップ (f) とを含む半導体集積回路装置の設計方法。

【請求項18】 各々ある機能を有する複数のIPを記憶装置に格納するステップ (a) と、

ある仕様を満たすための機能群構造を構築するステップ (b) と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ (c) と、

上記機能群中の各機能をデータベースのアーキテクチャ

モデル中のコンポーネントに割り当てるステップ (e) と、

上記コンポーネントのバスアクセスのための優先度を設定するステップ (f) と、

上記機能群構造の各機能が上記コンポーネントに割り当てられた状態で上記機能群構造の静的な性能解析を行なうステップ (g) とを含み、

上記静的な性能解析においては、上記各コンポーネントのアーキテクチャモデル中におけるバスアクセスの優先度をトグル数と見なして上記機能群構造の消費電力を計算することを特徴とする半導体集積回路装置の設計方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、複数の機能を有する回路を複数個配置した半導体集積回路装置の設計方法に関する。

【0002】

【従来の技術】 近年、複数のLSIを共通の基板上に形成したシステムLSIという概念が提起されており、システムLSIの設計手法としても各種の提案がなされている。特に、システムLSIの利点は、DRAMなどのメモリや、ロジックLSIや、高周波回路などのアナログ回路を1つの半導体装置内に収納して、多種、多機能の半導体装置を極めて高集積化して実現することができることである。

【0003】

【発明が解決しようとする課題】 ところで、上記従来のシステムLSIにおいては、システムLSIを設計する際に、IP、コアと呼ばれる設計資産を利用することが多い。そのとき、各コアは個別に設計されたもののので、これらを統合したときには必ずしも円滑な動作が得られないことがある。しかし、従来のコア又はIPは、その中身がブラックボックスとなっているので、円滑な動作を確保するシステムを設計するためには多大の手間と時間とを要しているのが現実である。

【0004】 本発明の目的は、IPの機能の同一性を図りつつIPの選択を柔軟に行なうための手段を講ずることにより、最適化が容易な半導体集積回路装置の設計方法の提供を図ることにある。

【0005】

【課題を解決するための手段】 本発明の第1の半導体集積回路装置の設計方法は、共通の機能を有する複数のIPを、複数の機能について記憶装置に格納するステップ (a) と、ある仕様を満たすための機能群構造を構築するステップ (b) と、上記記憶装置から上記機能群構造中の各機能について、当該機能を共通に有する複数のIPから1つのIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ (c) とを含んでいる。

10

20

30

40

50

【0006】この方法により、機能が互いに共通する複数のIPをLibに格納しておいて、機能ごとに適当なIPを選択することができるので、IPを代えても機能の同一性が保持される。したがって、従来大きかった設計変更の確率をできるだけ小さくすることができる。

【0007】上記各機能ごとに、当該機能の実現対象をデータベースのアーキテクチャモデル中のHWモデルとSWモデルとに割り当てるように分割するステップ

(e)とをさらに含むことにより、最適化された半導体集積回路装置を設計することができる。ができる。

【0008】上記ステップ(c)の後、上記IPの階層を機能ブロックレベルに展開するステップ(d)をさらに含み、上記ステップ(e)では、上記展開された各機能ブロックごとに、当該機能ブロックの実現対象をアーキテクチャモデル中のHWモデルとSWモデルとに割り当てることにより、より最適化された半導体集積回路装置を設計することができる。

【0009】本発明の第2の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のIPを記憶装置に格納するステップ(a)と、ある仕様を満たすための機能群構造を構築するステップ(b)と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ(c)と、上記IPを機能ブロックレベルに階層展開するステップ(d)と、上記階層展開された各機能ブロックごとに、当該機能ブロックの実現化対象をデータベースのアーキテクチャモデル中のHWモデルとSWモデルとに割り当てるように分割するステップ(e)とを含んでいる。

【0010】このように、IPを階層展開してからHWモデルとSWモデルとに分割することにより、IP内をHWモデルとSWモデルとに区別してマッピングを行なうことが可能となり、IP毎にマッピングする場合に比べて、より最適化されたシステムの設計が可能になる。

【0011】上記ステップ(e)では、機能ブロックのHWの部分から面積の重複を考慮した静的解析を行なうことにより、面積の計算精度の悪化を防止することができる。

【0012】上記ステップ(e)の後、割り当てられた上記機能群構造中の各機能ブロックのHWモデルとSWモデルとへの割り当てを切り換える動的解析を行なうステップ(f)をさらに含むことにより、解析精度の向上を図ることができる。

【0013】上記データベースには、HWの機能毎に当該HWと等価な機能を有するSWモデルを格納しておくことが好ましい。

【0014】上記ステップ(f)では、時間軸に沿って消費電力が常に上限値を超えないように各機能ブロックのHWモデルとSWモデルとへの割り当てを切り換えることができる。

【0015】上記ステップ(f)では、ある機能ブロックをHWモデルに割り当てたときのCPUの負荷率がある下限値よりも小さいときは、当該機能ブロックの割り当てをSWに切り換えることもできる。

【0016】記ステップ(e)では、メモリの消費電力を考慮した解析を行なうことを特徴とする半導体集積回路装置の設計方法。

【0017】本発明の第3の半導体装置集積回路装置の設計方法は、複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ(a)と、上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ(b)とを含み、上記ステップ(b)では、コンポーネント群に複数のバスと各バス間を接続するバスブリッジモデルとが含まれるように上記アーキテクチャを構築する方法である。

【0018】このような複数のバスを設けることにより、IPOSデバイスのような半導体配線基板を有する構造に適したアーキテクチャを構築することができる。

【0019】上記ステップ(b)では、上記バスブリッジモデルが、バス幅が互いに異なる2つのバス間をバス幅及びデータ転送速度を調整するように接続していることにより、当該バスブリッジモデルに接続される2つのバスのバス幅やデータ転送速度が互いに異なる場合などにおいてもデータの円滑な転送を図ることができる。

【0020】本発明の第4の半導体装置集積回路装置の設計方法は、複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ(a)と、上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ(b)と、上記仕様を満たすための機能群構造を構築するステップ(c)と、上記機能群構造の入力側及び出力側にテストベンチを付設するステップ(d)と、上記機能群構造中の機能及びテストベンチを上記アーキテクチャ中のコンポーネントに割り当てるステップ(e)とを含んでいる。

【0021】この方法により、テストベンチをHWにマッピングすることが可能になり、性能解析などにテストベンチの入出力に要する時間などのパラメータを含めることができる。

【0022】上記ステップ(a)では、上記複数のコンポーネントの中にI/Fモデルを含めておき、上記ステップ(b)では、上記コンポーネント群にI/Fモデルを含めるように上記アーキテクチャを構築することにより、入出力の信号を取り扱うI/Fモデルにおける信号の遅延や消費電力を考慮したシステムの最適化が可能になる。

【0023】上記ステップ(e)では、上記入力側のデ

ストベンチを上記アーキテクチャ中のI/Fモデルに割り当てることにより、従来、単に入力信号として扱われていたテストベンチがHWであるI/Fモデルにマッピングされるので、後の性能解析の際に遅延を考慮した処理時間の解析が可能になり、また、I/Fが動作することを考慮した消費電力の解析が可能になる。つまり、解析精度の向上を図ることができる。

【0024】上記ステップ(a)では、上記複数のコンポーネントの中にメモリを含めておき、上記ステップ

(b)では、上記コンポーネント群にメモリを含めるように上記アーキテクチャを構築し、上記ステップ(e)では、上記出力側のテストベンチを上記アーキテクチャモデル中のメモリに割り当てることもできる。

【0025】本発明の第5の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のIPを記憶装置に格納するステップ(a)と、ある仕様を満たすための機能群構造を構築するステップ(b)と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ(c)と、上記IPを機能ブロックレベルに階層展開するステップ(d)と、上記階層展開された各機能ブロックをアーキテクチャ中のコンポーネントに割り当てるステップ(e)と、上記機能ブロックから当該機能ブロックの出力を受ける他の機能ブロックまでの経路を、上記アーキテクチャ中のコンポーネント間の経路として記述する機能ブロック間テーブルを作成するステップ(f)とを含んでいる。

【0026】このように機能ブロック間パステーブルを作成することにより、動的な性能解析を精度よく行なうことができる。

【0027】本発明の第6の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のIPを記憶装置に格納するステップ(a)と、ある仕様を満たすための機能群構造を構築するステップ(b)と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ(c)と、上記機能群中の各機能をデータベースのアーキテクチャモデル中のコンポーネントに割り当てるステップ(e)と、上記コンポーネントのバスアクセスのための優先度を設定するステップ(f)と、上記機能群構造の各機能が上記コンポーネントに割り当てられた状態で上記機能群構造の静的な性能解析を行なうステップ(g)とを含み、上記静的な性能解析においては、上記各コンポーネントのアーキテクチャモデル中におけるバスアクセスの優先度をトグル数と見なし上記機能群構造の消費電力を計算する方法である。

【0028】優先度をトグル数の代わりに用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【0029】

【発明の実施の形態】-IPOSデバイスの基本的な構造-

ここで配線の断面積を大きく確保しつつ、多種、多機能のデバイスを内蔵した半導体装置を実現するための1つの手段として、配線層を有する半導体配線基板例えばシリコン配線基板(Super-Sub)上に、各種デバイスを内蔵したチップIPを搭載する構成がある。その場合、各チップIP内に設けられる回路(IC)は半導体装置の設計上IP(Intellectual Property)として扱うことができ、各種IPを半導体配線基板上に貼り合わせたものと考えることができる。つまり、半導体デバイス全体は、“IP On Super-Sub”であるので、シリコン配線基板とIP群とを備えた半導体デバイス全体を“IPOSデバイス”とする。

【0030】図1(a)、(b)、(c)は、IP(チップIP)群を搭載するための配線基板となるシリコン配線基板の平面図、シリコン配線基板上に搭載されるIP群の例を示す平面図、及びシリコン配線基板の断面図である。図1(a)、(b)に示すように、シリコン配線基板10上には各種IPを搭載するための複数の領域が設けられており、各領域には、例えば、Analog-IP、Logic-IP、CPU-IP、Flashメモリ-IP、SRAM-IP、DRAM-IP、I/O-IPなどの各種IP群がチップIPとして搭載可能となっている。図1(c)に示すように、シリコン配線基板10は、シリコン基板11と、シリコン基板10上に絶縁膜(図示せず)を挟んで設けられたグラウンドプレーン12と、グラウンドプレーン12の上に層間絶縁膜を挟んで設けられた第1配線層13と、第1配線層13の上に層間絶縁膜を挟んで設けられた第2配線層14と、第2配線層14の上にパッシベーション膜を挟んで設けられたパッド15とを備えている。パッド15、各配線層13、14及びグラウンドプレーン12間は、それぞれコンタクト(図示せず)を介して所望の部位で互いに接続されている。そして、各IPは、パッド15上に貼り付けられて、各IPが配線層13、14により互いにあるいはグラウンドプレーン12に電気的に接続される構造となっている。

【0031】シリコン配線基板10内の配線層13、14の寸法の制約は緩やかであり、数 μm 幅の配線をも設けることができるので、以下のような効果がある。経験的に、今までの半導体集積回路装置の微細化が進化した過程において、もっとも配線としての特性が良好であった世代の寸法を有する配線を設けることが可能となる。また、配線の電気インピーダンスを低減することができる。

【0032】そして、シリコン配線基板上のチップIPは多くの機能を有しているが、これに対しては2つの考え方があり、1つは、できるだけ多くの機能を使用する

という考え方であり、もう1つは、使用する機能を制限するという考え方である。つまり、ユーザの多様な要求に応えるためには、多種の使用方法に対応できる構成を有していることが好ましいが、反面、実際の使用に際してはいずれかの機能に限定する必要がある。つまり、この2つの相反する要求を満足させることにより、ユーザの多様な要求に応えつつ、少品種、大量生産に適したIPOSデバイスを構築することができるのである。

【0033】したがって、後に選択、制限、切り換え、設定などの処理を柔軟に行なうことを前提として、つまり、機能に関する処理の柔軟性を保持しつつ、多種の機能をハードウェア(HW)、ソフトウェア(SW)に分けて各IP(チップIP)に組み込むことが重要となる。

【0034】そこで、以下の実施形態においては、IPOSデバイスを例とした大規模なシステム(以下、「IPOSシステム」という)に適したHW/SW協調設計について説明する。ただし、本発明は、必ずしもIPOSシステムだけでなく、従来のいわゆるシステムLSIなどの大規模な集積回路システムの設計にも適用が可能である。

【0035】(実施形態)

—HW/SW協調設計全体の概略—

図2は、本発明の実施形態におけるHW/SW協調設計の概略的な手順を示すフロー図である。

【0036】ステップST1で、機能ライブラリ(機能Lib)やテストベンチを用いて機能設計を行ない、ステップST2で、IP機能Libを用いたIP機能設計を行なう。このステップST1、ST2では、半導体デバイスに搭載される各IPに要求される機能が例えばC言語により記述される。例えばJPEGの場合には、信号処理(アルゴリズム)だけが記載されているような状態である。

【0037】次に、ステップST3で、アーキテクチャモデルLibを用いてシステムのアーキテクチャ設計を行なう。システムのアーキテクチャモデルLibには、アーキテクチャモデルがシステムの設計に必要なHW、SWモデルとして存在しているので、このステップST3では、各モデルをバスに接続してシステム全体の概略的な構造を作り上げる。

【0038】次に、ステップST4で、各機能(機能ブロック)をアーキテクチャのコンポーネントに割り当てるマッピングを行なった後、ステップST5で、性能Libを用いて性能解析を行ない、その結果、不具合があればステップST3のアーキテクチャ設計、又はステップST4のマッピングに戻り、性能が満足されるまでこれらのステップを繰り返すループ処理を行なう。

【0039】次に、ステップST6、ST7で、それぞれIPハードウェアモデルLibを用いたHW設計と、例えばIPソフトウェアモデルLibを用いたSW設計

とを個別に行なってから、HW及びSWが詳細化されてくると、ステップST8で、I/Fモデルを用いたHW/SW協調検証を行なう。その後、ステップST9で、実チップを用いて実チップの機能を確認するための実チップ検証を行なう。

【0040】—機能設計及びIP機能設計—

図3は、機能設計を行なうステップST1と、IP機能設計を行なうステップST2との詳細な処理の内容を説明するための部分フロー図である。

【0041】ステップST1においては、以下の手順により、機能設計を行なう。機能Libには、f1、f2、f3、f4などの種々の機能(function)が格納されている。この機能f1、f2、f3、f4とは、例えば画像処理におけるMPEGKL、JPEG、Blue-toothなどの機能である。また、テストベンチLibには、tb1、tb2、tb3、tb4などの種々のテストベンチが格納されている。このテストベンチtb1、tb2、tb3、tb4は、機能設計が適正かどうかを判断するための与えられるデータである。例えば、図3のステップST1中に示すように、機能f1、f2、f3を組み合わせたデザインがある場合には、入力テストベンチとして、テストベンチからtb1を取り出して、これを入力データとして与える。また、出力データとして例えば画像出力のテストベンチtb2を与える。その結果、ここで作り上げたデザインが適正かどうかがある程度検証される。

【0042】次に、ステップST2におけるIP機能設計は以下の手順で行なわれる。IP機能Libには、各機能f1、f2、f3、f4ごとに、この機能を実現するためのいくつかのIPが格納されている。例えば、機能f1に対してf1-ip1～f1-ip4があり、機能f2に対してf2-ip1～f2-ip3があり、機能f3に対してf3-ip1、f3-ip2がある。そこで、おおよその見当をつけて、デザイン中の機能f1、f2、f3に対して用いるIPを指定する。例えば、機能f1に対してはf1-ip4を、機能f2に対してはf2-ip1を、機能f3に対してはf3-ip2を用いるというようにパラメータを指定する。

【0043】次に、ステップST1で作り上げられたデザイン中の各機能f1、f2、f3に、それぞれf1-ip4、f2-ip1、f3-ip2を当てはめる。つまり、IP機能の選択を行なう。この段階では、選出された各IPはいずれもステップST1における機能f1、f2、f3を有しているので、ステップST1で検証した結果と同じ結果が得られる。

【0044】このように、機能が互いに共通する複数のIPをLibに格納しておいて、機能ごとに適当なIPを選択することにより、IPを代えても機能の同一性が保持されるので、従来大きかった設計変更の確率をできるだけ小さくすることができる。

【0045】次に、IP機能をさらに詳細に個々の機能に分けるIP機能の階層展開を行なう。例えば、f1-ip4について階層展開をすると、f1-ip4-A（例えば逆コサイン変換）を行なってからf1-ip4-B（例えば量子化）を行ない、その後f1-ip4-C（例えばハフマン符号化）を行なうという機能になっていたとする。つまり、IPであるf1-ip4を機能ブロックf1-ip4-A、f1-ip4-B、f1-ip4-Cに展開するのである。同様に、f2-ip1、f3-ip2についても階層展開を行なって、f2-ip1についてはf2-ip1-Aという機能ブロックが、f3-ip2についてはf3-ip2-A、f3-ip2-Bという機能ブロックがそれぞれ得られたとする。そして、この階層展開された各機能ブロックをデザイン中の各機能f1、f2、f3に置換する。この階層展開されたIPの中身をHW/SW分割するのが本発明の1つの特徴である。

【0046】アーキテクチャ設計

次に、図4は、ステップST3における処理の内容を説明するための部分フロー図である。同図に示すように、アーキテクチャモデルLibには、CPU、メモリ、ASIC（ロジック回路）、BUS、バスブリッジモデル、I/Fモデルなどのコンポーネント(c)についての多くの種類が格納されている。そこで、このコンポーネントごとに各種のコンポーネントパラメータをもっているため、コンポーネントパラメータの値を与える。つまり、CPUについては動作周波数、キャッシュヒット率があり、DRAMについては動作周波数、ビット幅があり、ASICについては動作周波数、ビット幅があり、BUSについてはバスクロック、バス幅、転送レート(bus)，アービトレーション方法があり、バスブリッジモデルについてはバスブリッジ間遅延があり、I/FモデルとしてUARTを選択したときには入出力転送レート(bps)，入出力ビット幅、動作周波数、ビット幅がある。そして、各コンポーネントパラメータに具体的な値を入れる。そして、このパラメータ指定を行なってから、例えば図4に示すように、CPU1(c1)、DRAM1(c2)、ASIC1(c3)、ASIC2(c6)及びUART(c5)をBUS1、BUS2及びバスブリッジ1(c4)を介して接続した構造を有するアーキテクチャを作り上げる。

【0047】ここで、このアーキテクチャの特徴は、複数のバス(BUS1、BUS2)を備え、BUS1-BUS2間を接続するバスブリッジ1(c4)をコンポーネントとして設けている点である。バスは3つ以上あってもよく、このような複数のバスを設けることにより、半導体デバイスのような半導体配線基板を有する構造に適したアーキテクチャを構築することができる。複数のバスの例としては、標準バスに低速バス、高速バスなどを別途設けたものが考えられる。そして、バスブリッジ

モデルを設けることにより、当該バスブリッジモデルに接続される2つのバスのビット幅が互いに異なる場合などにおいてもデータの円滑な転送を図ることができる。バスブリッジモデルの例としては、16ビット幅のBUS（例えばバス1）のデータを8ビットずつ別のBUS（例えばBUS2）に送り込むバッファがある。また、アーキテクチャにおいて、I/FモデルをASICとは別に設けたことにより、入出力の信号を取り扱うI/Fモデルにおける信号の遅延や消費電力を考慮したシステムの最適化が可能になる。

【0048】マッピング

次に、図5は、ステップST4、ST5におけるマッピングと性能解析の概略とを説明するための部分フロー図である。

【0049】同図に示すように、ステップST4においては、ステップST2のIP機能設計によって得られたデザイン中の階層展開されたIPを、HWとSWとに割り当てる処理であるマッピングを行なう。例えば、テストベンチtb1は、HWとしてI/FモデルであるUARTに割り当てられる。また、テストベンチtb2は、HWとしてDRAM1に割り当てられる。また、階層展開されているf1-ip4中の機能ブロックf1-ip4-A(b11)はHWとしてASIC1に、機能ブロックf1-ip4-B(b12)、機能ブロックf1-ip4-C(b13)はSWとしてCPU1に、f2-ip1中の機能ブロックf2-ip1-A(b14)はSWとしてCPU1に、f2-ip2中の機能ブロックf2-ip2-A(b15)、機能ブロックf2-ip2-B(b16)はすべてHWとしてASIC2にそれぞれ割り当てられる。

【0050】ここで、このマッピングの特徴は、テストベンチをI/Fモデルにマッピングしたことである。従来、単に入力信号として扱われていたテストベンチをHWであるI/Fモデルにマッピングすることにより、後の性能解析の際に遅延を考慮した処理時間の解析が可能になり、また、I/Fモデルが動作することを考慮した消費電力の解析が可能になる。つまり、解析精度の向上を図ることができる。

【0051】また、このように、1つのIPをHWとSWとにわけてマッピングができるようになったのは、IP機能設計において、上述のようにIPを機能ブロックのレベルまで階層展開したからである。そして、マッピングに応じて、図5の左下に示すように、各機能とデザイン中のコンポーネントとのマッピング対応表が機能番号(a1~an)とともに作成される。また、例えば機能ブロックf1-ip4-A→機能ブロックf1-ip4-Bというバスは、ASIC1→BUS1→(DRAM1→)CPU1というデータのバスとなり、機能ブロックf1-ip4-B→機能ブロックf1-ip4-Cというバスは、CPU1→(DRAM1→)CPU2と

いうデータのパスとなること、ブロック間パステープルとして作成され、設計装置の記憶装置（図示せず）に格納される。なお、括弧内のコンポーネントはその部分が経由される場合もあり、経由されない場合もあることを示している。

【0052】ここで、このように機能ブロック間パステープルを作成することにより、後述するような動的な性能解析を精度よく行なうことができる。

【0053】次に、このようなマッピング結果を用いてステップST5における性能解析を行なう。このステップ

$$\begin{aligned} \text{Cost-tot} = & k1 (\text{Speed}_{\text{HW-tot}} + \text{Speed}_{\text{SW-tot}}) \\ & k2 (\text{Area}_{\text{HW-tot}} + \text{Area}_{\text{SW-tot}}) \\ & k3 (\text{Power}_{\text{HW-tot}} + \text{Power}_{\text{SW-tot}}) \end{aligned} \quad (1)$$

がある。ここで、 $k1$ 、 $k2$ 、 $k3$ は重み係数であり、 $\text{Speed}_{\text{HW-tot}}$ はHW全体の処理速度であり、 $\text{Speed}_{\text{SW-tot}}$ はSW全体の処理速度であり、 $\text{Area}_{\text{HW-tot}}$ はHW全体の面積であり、 $\text{Area}_{\text{SW-tot}}$ はSW全体の面積であり、 $\text{Power}_{\text{HW-tot}}$ はHW全体の消費電力であり、 $\text{Power}_{\text{SW-tot}}$ はSW全体の消費電力である。つまり、この例では、パラメータである処理速度ができるだけ大きく、面積ができるだけ小さく、消費電力ができるだけ小さくなるようにシステムを構築したいと考えていることにな

$$\begin{aligned} \text{Speed}_{\text{HW-tot}} (f_{b11} \sim f_{b16}, f_{bus}, R_{bus}, P_{b11} \sim P_{b16}) \\ = \text{Speed}_{\text{HW-b11}} (f_{o1}, f_{bus}, R_{bus}, R_{o1}) + \\ \dots + \text{Speed}_{\text{HW-b16}} (f_{o1}, f_{bus}, R_{bus}, P_{o1}) \end{aligned} \quad (2)$$

のように行なわれる。ここで、式(2)において、図5に示すように、マッピングしたときの機能ブロック番号を**b11**～**b16**とし、コンポーネント番号を**c1**～**c6**とする。そして、 f は動作周波数であり、 R はバスの転送レートであり、 P は機能のバスに対する優先順位（優先度）（0～1の値）を表している。この優先順位★

$$\begin{aligned} \text{Speed}_{\text{HW-b11}} (f_{o1}, f_{bus}, R_{bus}, P_{o1}) \\ = 1 / (f_{o1} \times f_{bus} \times R_{bus} \times P_{o1}) \end{aligned} \quad (3)$$

のように求められる。つまり、処理速度はできる限り小さい方が好ましいことから、動作周波数 f 、転送レート R 、優先順位 P の逆数からコスト関数（Cost-tot）のうち処理速度Speedの成分を計算する。

$$\begin{aligned} \text{Speed}_{\text{SW-tot}} (St_{\text{CPU-o1}} \sim St_{\text{CPU-o6}}, f_{\text{CPU}}) \\ = \text{Speed}_{\text{SW-b11}} (St_{\text{CPU-o1}}, f_{\text{CPU}}) + \\ \dots + \text{Speed}_{\text{SW-b16}} (St_{\text{CPU-o6}}, f_{\text{CPU}}) \end{aligned} \quad (4)$$

により求められる。この計算の考え方は以下の通りである。機能ブロックをfunctionによって表すと、処理はC言語によって記述することができ、C言語で記述すればアセンブリ言語にコンパイルすることが可能となる。したがって、アセンブラのステップ数で処理速度◆

$$\text{Speed}_{\text{SW-b11}} (St_{\text{CPU-o1}}, f_{\text{CPU}}) = St_{\text{CPU-o1}} \times f_{\text{CPU}} \quad (5)$$

のように求められる。このステップ数が小さいほど速度が大きいので、式(5)の数値がなるべく小さいことが望ましいことになる。

$$\text{Area}_{\text{HW-tot}} = \text{Area}_{\text{HW-b1} \sim \text{b16}}$$

＊ブにおいては、静的解析と動的解析とを行なってもよいし、静的解析及び動的解析のうちいずれか一方のみを行なってもよい。

【0054】－性能解析（静的解析）－

性能解析のうち静的解析は、例えば所望のパラメータを変数としてシステム全体のコスト関数（Cost-tot）を定義して、このコスト関数（Cost-tot）を求める。そして、コスト関数（Cost-tot）が最小又はある上限値以下になるかを目標として設計を進める。このコスト関数（Cost-tot）の一例として、例えば下記式（1）

※る。そこで、コスト関数（Cost-tot）は小さいほどよいとする。例えば、各種のマッピングMAP1、MAP2、MAP3を行なった結果、コスト関数（Cost-tot）が図6に示す値になったとする。このときには、もっとも値が小さいマッピングMAP1が最適であると判断する。

【0055】ここで、例えば処理速度に関する計算は、マッピング結果を用いて、下記式（2）

★ P は、バスに接続されるすべてのコンポーネントの優先度を合計すると1になるように設定されている。

【0056】すると、式（2）中の例えば機能ブロック**b11**（ $f1-i p 4-A$ ）の処理速度は、下記式（3）

☆【0057】次に、式（1）中のSWについての処理速度成分 $\text{Speed}_{\text{SW-tot}}$ は、マッピングのブロック間パステープルに基づいて、下記式（4）

◆を演算することができるからである。

【0058】ここで、すると、式（4）中の例えば機能ブロック**b11**（ $f1-i p 4-A$ ）の処理速度は、下記式（5）

【0059】次に、面積に関する計算は、マッピング結果を用いて、下記式（6）

$$+ \text{Area}_{\text{HW-IP1}} + \text{Area}_{\text{HW-IP2}}$$

のように行なわれる。つまり、IP機能を選択した後でIP機能の階層展開を行なう前に使用される各IP ($f1-ip4$, $f2-ip1$, $f3-ip2$) の情報があるので、これからHW部分の面積を求めることができる。

【0060】次に、機能ブロックをコンパイルすると、*

$$\text{Area}_{\text{SW-tot}} = (\text{Msize}_{\text{SW-111}} + \dots + \text{Msize}_{\text{SW-b11}}) \times \text{Area}_{\text{DRAM}} \quad (7)$$

により、求めることができる。

【0061】このように、IP内が機能ブロックに分割されたことにより、機能ブロックのSWの部分からメモリサイズを計算すると、各機能ブロック間で面積を重複※

$$\text{Power}_{\text{HW-tot}} = \text{Power}_{\text{HW-b1}} + \dots + \text{Power}_{\text{HW-b11}} \quad (8)$$

のように行なわれる。すると、式(8)中の例えば機能ブロックb11 ($f1-ip4-A$) の消費電力は、下★

$$\begin{aligned} \text{Power}_{\text{HW-b11}}(f_{c3}, V_{c3}) \\ = \alpha_{b11} \times f_{c3} \times (V_{c3})^2 \times P_{c3} \end{aligned} \quad (9)$$

により、表される。ただし、 α は機能ブロックb11のパワー係数でIP性能Libから抽出されるもの、 f_{c3} はコンポーネントc3 (ASIC1) の動作周波数、 V_{c3} はコンポーネントc3 (ASIC1) の電圧、 P_{c3} はコンポーネントc3 (ASIC1) の優先順位(優先 ☆

$$\text{Power}_{\text{SW-tot}} = \text{Power}_{\text{SW-CPU-tot}} + \text{Power}_{\text{SW-MEM-tot}} \quad (10)$$

により求められる。式(10)の第1項は、

$$\text{Power}_{\text{SW-CPU-tot}} = \alpha_{\text{CPU}} \times f_{\text{CPU}} \times \text{Load}_{\text{CPU}} \quad (11)$$

となる。ここで、 α_{CPU} はCPUのパワー係数でIP性能Libから抽出されるもの、 Load_{CPU} はCPUロード(0~1)であって、「SWにマッピングされた機能ブロックのメモリサイズの合計(bit)」を「すべての◆

$$\text{Power}_{\text{SW-MEM-tot}} = \alpha_{\text{DRAM}} \times f_{\text{DRAM}} \times (V_{\text{DRAM}})^2 \times P_{\text{DRAM}} \quad (12)$$

となる。 α_{DRAM} はDRAMのパワー係数でIP性能Libから抽出されるもの、 f_{DRAM} はDRAMの動作周波数(Hz)、 V_{DRAM} はDRAMの電圧、 P_{DRAM} はDRAMの優先順位を示している。この優先度を用いることで、トグル数がわからない静的解析においても、ある程度の精度で消費電力が求まることになる。

【0065】そして、以上の計算を行なって上で、式(1)から最終的に決定されるコスト関数(Cost-tot)が最小になるか、下限値以下になったときに静的解析に関する限り、設計されたシステムが適切であると判断する。

【0066】-性能解析(動的解析)-

動的解析においては、面積Areaについては静的解析と同じであるので、処理速度Speedと消費電力Powerのみを求める。動的解析においては、設計システムに実際にデータを入力させて動作させるシミュレーションを行なう。したがって、実行時間(Time(sec))や、各機能ブロックの入出力データ、トグル率(5)がシミュレーションから得られる。

【0067】そこで、動的解析においては、基本的に

(6)

* アセンブラでメモリサイズがわかるので、SWのメモリの部分をすべて加算しておいて、それにエリアDRAMというデータベースを用いてDRAMの1ビット当たりの面積(単価)を計算する。それに、メモリサイズを乗じると、下記式(7)

※して計算する場合があるので、これを考慮して面積の計算精度の悪化を防止するのである。

【0062】次に、HW部分の消費電力に関する計算は、マッピング結果を用いて、下記式(8)

★記式(9)

☆度)を示している。この優先度をトグル数の代わりに用いることで、トグル数がわからない静的解析においても、ある程度の精度で消費電力が求まることになる。

【0063】また、SW部分の消費電力に関する計算は、下記式(10)

◆機能ブロックをSWにマッピングしたときのメモリサイズの合計(bit)」で除したものである。

【0064】一方、式(10)の第2項は、

は、この得られた要素を上記式(1)~(12)に代入することにより、コスト関数(Cost-tot)の値を求めることができる。その際、式(1)における($\text{Speed}_{\text{HW-tot}} + \text{Speed}_{\text{SW-tot}}$)は、シミュレーションで得られた時間Timeに置き換えることができ、式(9)における各コンポーネントの優先順位Pはシミュレーションで得られた各コンポーネントのトグル数で置き換えることができる。そして、式(11)におけるCPUロード Load_{CPU} は、実際のシミュレーションから求まる平均ロード(5)を用いる。

【0068】さらに、式(12)におけるDRAMの優先順位P(優先度)はシミュレーションで得られたDRAMのトグル数で置き換えることができる。

【0069】-HW/SW分割(静的分割)-

次に、ステップST4、ST5と共に行なわれるIP内のHW/SW分割について説明する。HW/SWには静的分割と動的分割とがあるが、まず、静的な分割について説明する。

【0070】図7は、静的なHW/SW分割を説明するための図である。同図に示すように、半導体デバイスな

どのシステムに利用されるハードIPには、機能A、機能B、機能C、機能Dがあり、かつ、これらの機能のON・OFFを設定するための設定レジスタがある。ただし、HW/SW分割における「機能」とは本実施形態における「機能ブロック」を意味している。一方、IP性能Libには、各機能A、機能B、機能C、機能DについてのSWモデルが例えばC/C++言語によって記述されて格納している。このとき、各機能をHWとSWとによって動作させた結果、1つの機能を変更、修正、削除する必要が生じる場合がある。ここで、例えば、設定レジスタ内の機能DをOFFにすることで、例えば機能Dへのクロックの供給を停止するなどの制御を行なうと、機能Dが動作しないようにすることができる。これにより、無駄な電力の消費を抑制して消費電力の低減を図ることができる。また、各機能A～D間の接続を変更することもできる。

【0071】さらに、例えば1つの機能DをHWからIP性能Libに用意されたSWモデルに切り換えたり、用意しているLib内のSWモデルを修正することもできる。つまり、いずれかのLib内に各機能と等価なSWモデルを準備しておくことにより、1つのIP内をHW/SW分割することができ、ハードIPの機能が不足したときにも設計を進めることができる。また、SWの機能を修正、追加することにより、システムの変更に柔軟に対応することができる。

【0072】—動的なHW/SW分割—

次に、動的なHW/SW分割について説明する。図7に示すHW/SW分割は、HW、SWの設定を行なった後は、原則としてその後の修正は行なわない。それに対し、動的なHW/SW分割、つまりHW/SW動作の切り換えにおいては、機能（機能ブロック）が動作しながら柔軟に変更、追加、修正を受けることができる。このような動的なHW/SW分割の例として、以下、第1～第3の具体例について説明する。

【0073】1. 第1の具体例

第1の具体例では、図7に示すIPにおいて、各機能のON、OFFをSWによって制御し、かつ、停止させる場合はIP性能Libから同等のSWモデルを呼び出してこれを用いる。設定レジスタへの制御信号により、各機能を動作させながらHW/SWの切り換えを行なうことができる。

【0074】2. 第2の具体例

図8(a)、(b)は、動的なHW/SWの切り換えにおける消費電力Powerのタイムチャート図である。また、図9は、このHW/SWの切り換え制御のためのプログラムの例をC言語で示す図である。図9に示すように、設定レジスタの制御により、ハードIPの機能Cを動作させた場合に、図8(a)に示すように、IPの消費電力Powerが消費電力の上限値MAXpowerを越える時があるとする。動的解析からどの機能が動作している

かが時間軸に応じてわかるので、あるレジスタに各機能の消費電力の合計を計算しながら書き込んでおいて、その値が上限値MAXpowerを越えるときを検出すればよい。そして、IPの消費電力Powerが消費電力の上限値MAXpowerを越えると、図9に示すように、設定レジスタの制御によりハードIPの機能CをOFFにして、代わりにSWモデルからfunc-Cを呼び出してこちらを動作させる。これにより、図8(b)に示すように、消費電力Powerが上限値MAXpowerを越えることがなくなり、所望の制約条件の下での作動を確保することができる。つまり、この例では、当該IPの機能A、Cが動作する期間には機能CがSWによって実現にされるが、機能A、B、Cが動作する期間には機能CがHWによって実現されるように設計することになる。

【0075】3. 第3の具体例

図10(a)、(b)は、動的なHW/SWの切り換えにおける処理速度Speedの消費電力依存性を示す図である。また、図11は、このHW/SWの切り換え制御のためのプログラムの例をC言語で示す図である。図11に示すように、(注：不等号の向きが逆のように思われましたので変更いたしました。どうか) CPUの設定レジスタの制御により、CPUのロード(%)がある値IDLEよりも大きいときには機能CをSWにより実行し、CPUのロード(%)がある値IDLE以下のときに機能CをHWにより実行するよう制御する。つまり、機能Cを常にHWで実行すると、図10(a)に示すように処理時間Timeが上限値MAXtimeを越える事態が発生する。そこで、レジスタにCPUのロード(%)を検知しながら書き込んでおいて、CPUのロード(%)が下限値IDLEを越えるときを検出しながら、CPUのロード(%)がある値IDLE以下のときには機能CをHWにより実行する。そして、CPUのロード(%)が下限値IDLEを越えると、設定レジスタの制御によりハードIPの機能CをOFFにして、代わりにSWモデルからfunc-Cを呼び出してSWを動作させる。このような制御を行なうことにより、図10(b)に示すように、処理時間Timeが上限値MAXtimeを越えることがなくなり、所望の制約条件の下での作動を確保することができる。

【0076】—HW/SW協調検証—

次に、ステップST8の協調検証について説明する。上述のステップST4のマッピング処理、ステップST5の性能解析処理に基づいて最終的に決定されたHW、SWの構成に従い、ステップST6、ST7で、HW設計とSW設計とを行なうと、HWモデルとSWモデルとをそれぞれ作成する。HWモデルを作成する際にはIPハードウェアモデルLib(BCA、RTL又はゲートレベルで記述)を利用し、SWモデルを作成する際にはIPソフトウェアモデルLib(C言語、C++言語又はアセンブラで記述)を利用する。

【0077】次に、ステップST8では、ステップST6、ST7で作成したHWモデル及びSWモデルを利用して、HW/SW協調検証を行なうが、そのためにはHW-SW間のI/Fモデルが必要となる。そこで、以下の手順により、I/Fモデルモデルを作成する。

【0078】まず、図13に示すようなメモリマップを作成する。このメモリマップは、SWにプログラムをコンパイルしたものである。HWは、レジスタをメモリにマッピングする。

【0079】そして、各機能ブロックのアドレスを決定し、各機能ブロックのアドレステーブルをデータベース化して、メモリマップDBに登録する。そして、図14に示すように、メモリマップDBに登録されたHWのレジスタとアドレスとの対応表をとりだして、アドレスデコーダを作成する。

【0080】バスコントローラは、各HWブロックのP（優先度）にしたがって、各バス毎のアービトレーション方法ごとに用意したHDLテンプレートを生成する。すなわち、バスコントローラは、各ハードウェアのレジスタアドレスをメモリマップDBから抽出して、関数を作成し、これをドライバ関数テンプレートとしてSWモデルに格納する。

【0081】図15は、基本ハードウェアアクセス用のドライバ関数テンプレートの一例を示す図である。このようなメモリマップDB、アドレスデコーダ、関数テンプレートを利用することにより、設計されたシステムのHWによってSWを動作させたり、システムのSWによってHWを動作させることが可能になり、HW/SW協調検証を行なうことができる。

【0082】

【発明の効果】本発明により、IPの機能に着目してより最適化された半導体集積回路装置を形成するための設計が可能になった。

【図面の簡単な説明】

【図1】(a)、(b)、(c)は、IP群を搭載するための配線基板となるシリコン配線基板の平面図、シリコン配線基板上に搭載されるIP群の例を示す平面図、及びシリコン配線基板の断面図である。

【図6】

	MAP1	MAP2	MAP3
Cost-tot	80	100	103

* 【図2】本発明の実施形態におけるHW/SW協調設計の概略的な手順を示すフロー図である。

【図3】機能設計とIP機能設計との詳細な処理の内容を説明するための部分フロー図である。

【図4】システムアーキテクチャ設計における処理の内容を説明するための部分フロー図である。

【図5】マッピングと性能解析の概略とを説明するための部分フロー図である。

【図6】各種のマッピングを行なった結果のコスト関数（Cost-tot）の値の例を示す図である。

【図7】静的なHW/SW分割を説明するための図である。

【図8】(a)、(b)は、動的なHW/SWの切り換えにおける消費電力Powerのタイムチャート図である。

【図9】消費電力を考慮したHW/SWの切り換え制御のためのプログラムの例をC言語で示す図である。

【図10】(a)、(b)は、動的なHW/SWの切り換えにおける処理速度Speedの消費電力依存性を示す図である。

【図11】処理速度を考慮したHW/SWの切り換え制御のためのプログラムの例をC言語で示す図である。

【図12】(a)、(b)は、第2の実施形態の第5の具体例における機能の設定に関する半導体デバイスの構成を示す斜視図及びブロック回路図である。

【図13】SWにプログラムをコンパイルして得られたメモリマップを示す図である。

【図14】メモリマップDBに登録されたHWのレジスタとアドレスとの対応表をとりだして、アドレスデコーダを作成する処理を示す図である。

【図15】基本ハードウェアアクセス用のドライバ関数テンプレートの一例を示す図である。

【符号の説明】

- 10 シリコン配線基板
- 11 シリコン基板
- 12 グランドプレーン
- 13 第1配線層
- 14 第2配線層
- 15 パッド

【図9】

```

if(Power(now) > MAXpower)
{
    regC=ON;
} else{
    regC=OFF;
    func_C( );
}

```

【図11】

```

if(Load cpu > IDLE)
{
    regC=OFF;
    func_C( );
} else{
    regC=ON;
}

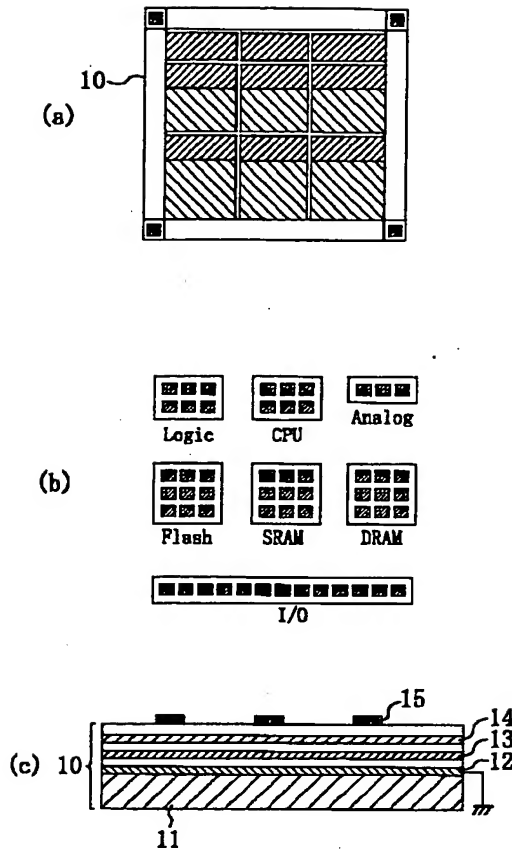
```

【図13】

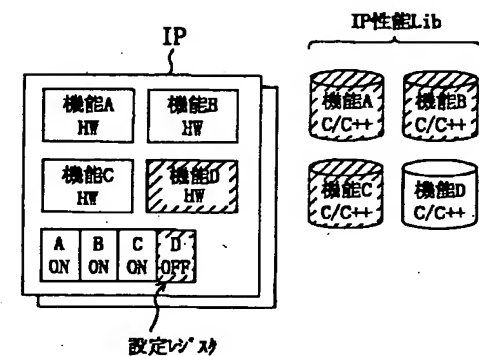
メモリマップ	
アドレス	
f1-ip4-B	
f1-ip4-C	
f2-ip1-A	

Adr

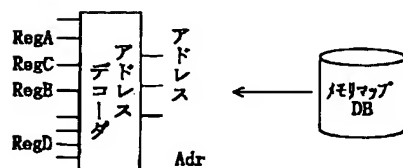
【図1】



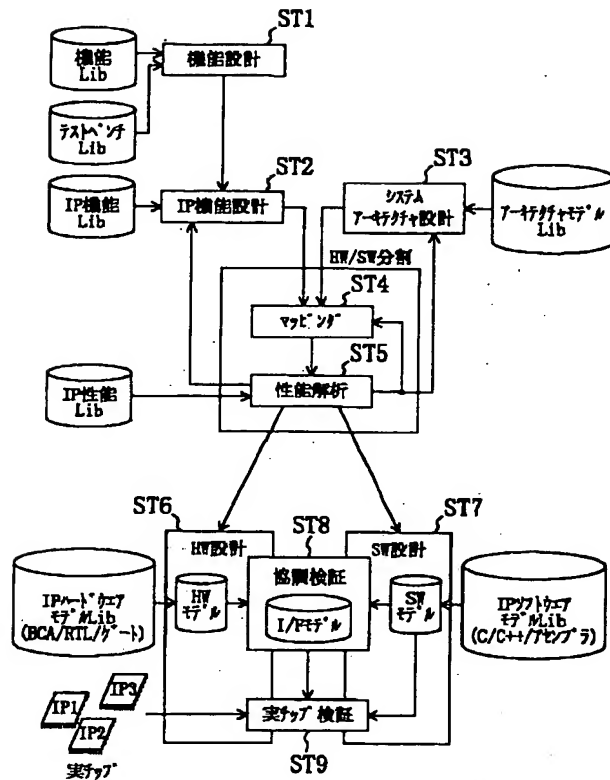
【図7】



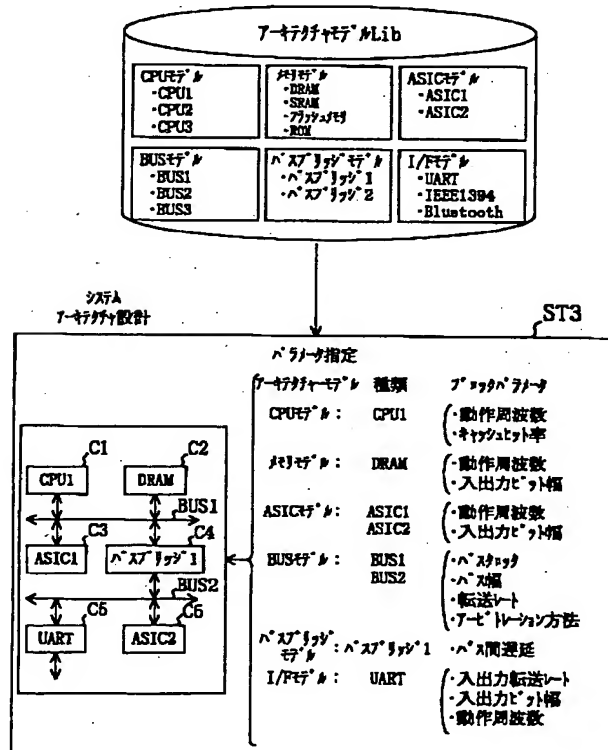
【図14】



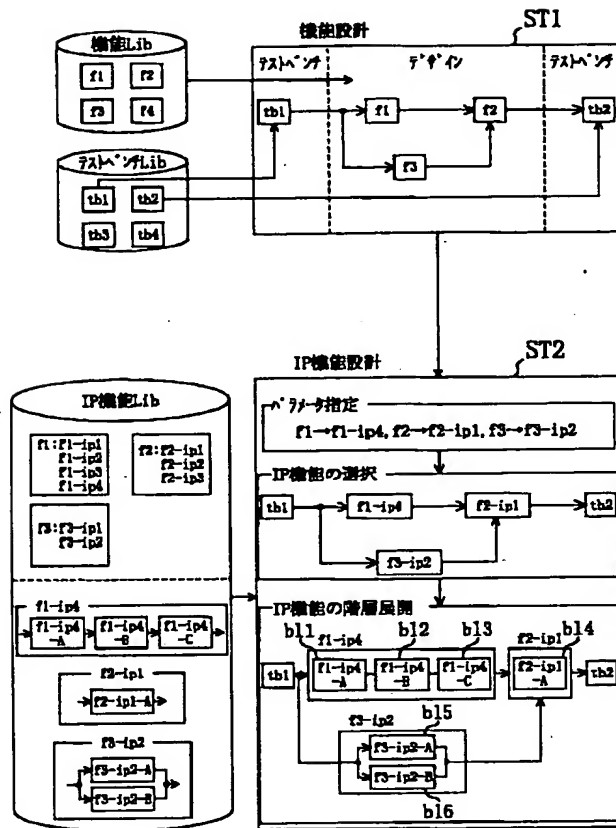
【図2】



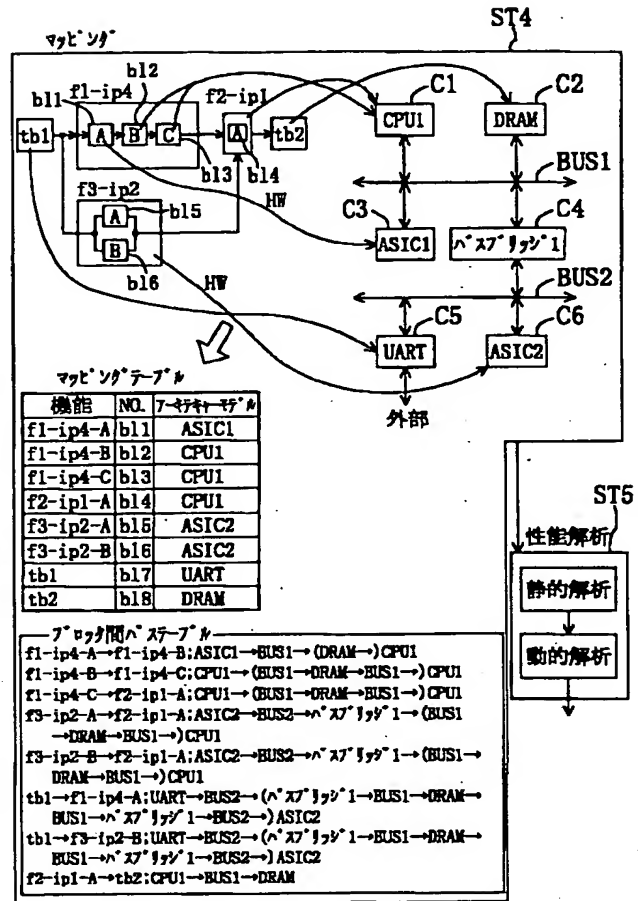
【図4】



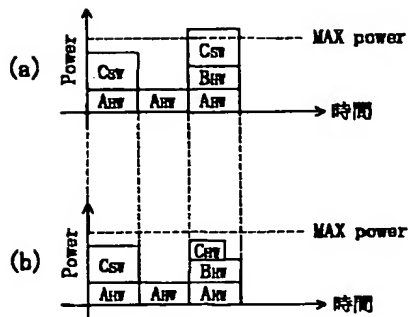
【図 3】



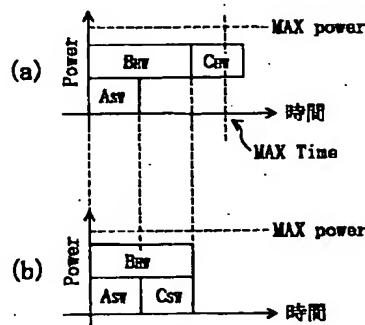
【図 5】



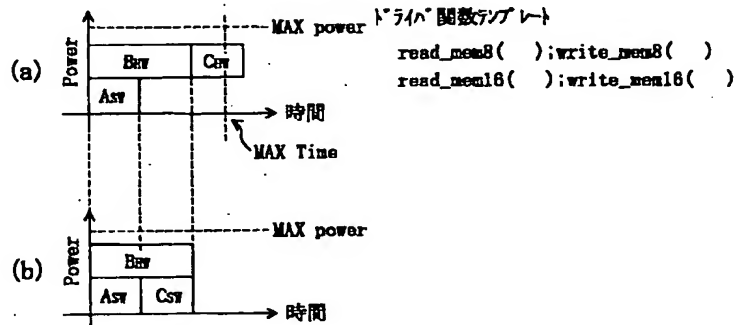
【図 8】



【図 10】



【図 15】



The diagram illustrates a system architecture for hardware and software development, organized into three main stages: ST6 (HW設計), ST8 (協調検証), and ST7 (SW設計).

- ST6 (HW設計):** Contains an "HW設計" block and an "IPポートライブラリ (BCA/RTL/ターゲット)" database. It also receives input from "IP1", "IP2", and "IP3" (実チップ).
- ST7 (SW設計):** Contains an "SW設計" block and an "IPソフトウェアライブラリ (C/C++/ターゲット)" database.
- ST8 (協調検証):** A central block containing an "I/Fポート" database. It receives input from the "HW設計" block in ST6 and the "SW設計" block in ST7.
- 実チップ検証:** A block below ST8 that receives input from the "I/Fポート" database in ST8 and the "実チップ" (IP1, IP2, IP3).
- Bottom Section:** A large "I/Fポート" database connected to the "実チップ検証" block via dashed lines. This database is further connected to "HWポート" and "SWポート" databases. It also contains a table with the following structure:

テストユーザ	ポートライブラリ
ハードウェア	ソフトウェア

 The table is labeled "ハードウェア (RTL/ターゲット)" and "ソフトウェア (C/C++/ターゲット)".

上記ステップ(e)では、機能ブロックのHWの部分から面積の重複を考慮した静的解析を行なうことを特徴とする半導体集積回路装置の設計方法。

【請求項6】 請求項4記載の半導体集積回路装置の設計方法において、

上記ステップ(e)の後、割り当てられた上記機能群構造中の各機能ブロックのHWモデルとSWモデルとへの割り当てを決定するための動的性能解析を行なうステップ(f)をさらに含むことを特徴とする半導体集積回路装置の設計方法。

【請求項7】 請求項6記載の半導体集積回路装置の設計方法において、

上記データベースには、HWの機能毎に当該HWと等価な機能を有するSWモデルを格納しておくことを特徴とする半導体集積回路装置の設計方法。

【請求項8】 請求項7記載の半導体集積回路装置の設計方法において、

上記ステップ(f)では、時間軸に沿って消費電力が常に上限値を超えないように各機能ブロックのHWモデルとSWモデルとへの割り当てを切り換えることを特徴とする半導体集積回路装置の設計方法。

【請求項9】 請求項6記載の半導体集積回路装置の設計方法において、

上記ステップ(f)では、ある機能ブロックをHWに割り当てたときのCPUの負荷率がある下限値よりも小さいときは、当該機能ブロックの割り当てをSWに切り換えることを特徴とする半導体集積回路装置の設計方法。

【請求項10】 請求項5～9のうちいずれか1つに記載の半導体集積回路装置の設計方法において、

上記ステップ(e)では、メモリの消費電力を考慮した解析を行なうことを特徴とする半導体集積回路装置の設計方法。

【請求項11】 複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ

(a)と、

上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ

(b)とを含む、

上記ステップ(b)では、コンポーネント群に複数のバスと各バス間を接続するバスブリッジモデルとが含まれるように上記アーキテクチャを構築することを特徴とする半導体集積回路装置の設計方法。

【請求項12】 請求項11記載の半導体集積回路装置の設計方法において、

上記ステップ(b)では、上記バスブリッジモデルが、バス幅が互いに異なる2つのバス間をバス幅及びデータ転送速度を調整するように接続していることを特徴とする半導体集積回路装置の設計方法。

【請求項13】 複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ

(a)と、

上記記憶装置からアーキテクチャモデルをコンポーネン

ト群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ

(b)と、

上記仕様を満たすための機能群構造を構築するステップ(c)と、

上記機能群構造の入力側及び出力側にテストベンチを付設するステップ(d)と、

上記機能群構造中の機能及びテストベンチを上記アーキテクチャ中のコンポーネントに割り当てるステップ(e)とを含む半導体集積回路装置の設計方法。

【請求項14】 請求項13記載の半導体集積回路装置の設計方法において、

上記ステップ(a)では、上記複数のコンポーネントの中にI/Fモデルを含めておき、

上記ステップ(b)では、上記コンポーネント群にI/Fモデルを含めるように上記アーキテクチャを構築することを特徴とする半導体集積回路装置の設計方法。

【請求項15】 請求項14記載の半導体集積回路装置の設計方法において、

上記ステップ(e)では、上記入力側又は出力側のテストベンチを上記アーキテクチャ中のI/Fモデルに割り当てることを特徴とする半導体集積回路装置の設計方法。

【請求項16】 請求項13～15のうちいずれか1つに記載の半導体集積回路装置の設計方法において、

上記ステップ(a)では、上記複数のコンポーネントの中にメモリを含めておき、

上記ステップ(b)では、上記コンポーネント群にメモリを含めるように上記アーキテクチャを構築し、

上記ステップ(e)では、上記入力側又は出力側のテストベンチを上記アーキテクチャ中のメモリに割り当てることを特徴とする半導体集積回路装置の設計方法。

【請求項17】 各々ある機能を有する複数のIPを記憶装置に格納するステップ(a)と、

ある仕様を満たすための機能群構造を構築するステップ(b)と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ(c)と、

上記IPを機能ブロックレベルに階層展開するステップ(d)と、

上記階層展開された各機能ブロックをアーキテクチャモデル中のコンポーネントに割り当てるステップ(e)

と、

上記機能ブロックから当該機能ブロックの出力を受ける他の機能ブロックまでの経路を、上記アーキテクチャモデル中のコンポーネント間の経路として記述する機能ブロック間テーブルを作成するステップ(f)とを含む半導体集積回路装置の設計方法。

【請求項18】 各々ある機能を有する複数のIPを記

憶装置に格納するステップ(a)と、ある仕様を満たすための機能群構造を構築するステップ(b)と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ(c)と、上記機能群中の各機能をデータベースのアーキテクチャモデル中のコンポーネントに割り当てるステップ(e)と、

上記コンポーネントのバスアクセスのための優先度を設定するステップ(f)と、

上記機能群構造の各機能が上記コンポーネントに割り当てられた状態で上記機能群構造の静的な性能解析を行なうステップ(g)とを含み、

上記静的な性能解析においては、上記各コンポーネントのアーキテクチャモデル中におけるバスアクセスの優先度をトグル数と見なして上記機能群構造の消費電力を計算することを特徴とする半導体集積回路装置の設計方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、複数の機能を有する回路を複数個配置した半導体集積回路装置の設計方法に関する。

【0002】

【従来の技術】近年、複数のLSIを共通の基板上に形成したシステムLSIという概念が提起されており、システムLSIの設計手法としても各種の提案がなされている。特に、システムLSIの利点は、DRAMなどのメモリや、ロジックLSIや、高周波回路などのアナログ回路を1つの半導体装置内に収納して、多種、多機能の半導体装置を極めて高集積化して実現することができることである。

【0003】

【発明が解決しようとする課題】ところで、上記従来のシステムLSIにおいては、システムLSIを設計する際に、IP、コアと呼ばれる設計資産を利用することが多い。そのとき、各コアは個別に設計されたものなので、これらを統合したときには必ずしも円滑な動作が得られないことがある。しかし、従来のコア又はIPは、その中身がブラックボックスとなっているので、円滑な動作を確保するシステムを設計するためには多大の手間と時間とを要しているのが現実である。

【0004】本発明の目的は、IPの機能の同一性を図りつつIPの選択を柔軟に行なうための手段を講ずることにより、最適化が容易な半導体集積回路装置の設計方法の提供を図ることにある。

【0005】

【課題を解決するための手段】本発明の第1の半導体集積回路装置の設計方法は、共通の機能を有する複数のIP

Pを、複数の機能について記憶装置に格納するステップ(a)と、ある仕様を満たすための機能群構造を構築するステップ(b)と、上記記憶装置から上記機能群構造中の各機能について、当該機能を共通に有する複数のIPから1つのIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ(c)とを含んでいる。

【0006】この方法により、機能が互いに共通する複数のIPをLibに格納しておいて、機能ごとに適当なIPを選択することができるので、IPを代えても機能の同一性が保持される。したがって、従来大きかった設計変更の確率をできるだけ小さくすることができる。

【0007】上記各機能ごとに、当該機能の実現対象をデータベースのアーキテクチャモデル中のHWモデルとSWモデルとに割り当てるように分割するステップ

(e)とをさらに含むことにより、最適化された半導体集積回路装置を設計することができる。ができる。

【0008】上記ステップ(c)の後、上記IPの階層を機能ブロックレベルに展開するステップ(d)をさらに含み、上記ステップ(e)では、上記展開された各機能ブロックごとに、当該機能ブロックの実現対象をアーキテクチャモデル中のHWモデルとSWモデルとに割り当てることにより、より最適化された半導体集積回路装置を設計することができる。

【0009】本発明の第2の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のIPを記憶装置に格納するステップ(a)と、ある仕様を満たすための機能群構造を構築するステップ(b)と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ(c)と、上記IPを機能ブロックレベルに階層展開するステップ(d)と、上記階層展開された各機能ブロックごとに、当該機能ブロックの実現化対象をデータベースのアーキテクチャモデル中のHWモデルとSWモデルとに割り当てるように分割するステップ(e)とを含んでいる。

【0010】このように、IPを階層展開してからHWモデルとSWモデルとに分割することにより、IP内をHWモデルとSWモデルとに区別してマッピングを行なうことが可能となり、IP毎にマッピングする場合に比べて、より最適化されたシステムの設計が可能になる。

【0011】上記ステップ(e)では、機能ブロックのHWの部分から面積の重複を考慮した静的解析を行なうことにより、面積の計算精度の悪化を防止することができる。

【0012】上記ステップ(e)の後、割り当てられた上記機能群構造中の各機能ブロックのHWモデルとSWモデルとへの割り当てを切り換える動的な性能解析を行なうステップ(f)をさらに含むことにより、解析精度の向上を図ることができる。

【0013】上記データベースには、HWの機能毎に当該HWと等価な機能を有するSWモデルを格納しておくことが好ましい。

【0014】上記ステップ(f)では、時間軸に沿って消費電力が常に上限値を超えないように各機能ブロックのHWモデルとSWモデルとへの割り当てを切り換えることができる。

【0015】上記ステップ(f)では、ある機能ブロックをHWモデルに割り当てたときのCPUの負荷率がある下限値よりも小さいときは、当該機能ブロックの割り当てをSWに切り換えることもできる。

【0016】上記ステップ(e)では、メモリの消費電力を考慮した解析を行なう。

【0017】本発明の第3の半導体装置集積回路装置の設計方法は、複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ(a)と、上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ(b)とを含み、上記ステップ(b)では、コンポーネント群に複数のバスと各バス間を接続するバスブリッジモデルとが含まれるように上記アーキテクチャを構築する方法である。

【0018】このような複数のバスを設けることにより、IPOSデバイスのような半導体配線基板を有する構造に適したアーキテクチャを構築することができる。

【0019】上記ステップ(b)では、上記バスブリッジモデルが、バス幅が互いに異なる2つのバス間をバス幅及びデータ転送速度を調整するように接続していることにより、当該バスブリッジモデルに接続される2つのバスのバス幅やデータ転送速度が互いに異なる場合などにおいてもデータの円滑な転送を図ることができる。

【0020】本発明の第4の半導体装置集積回路装置の設計方法は、複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ(a)と、上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ(b)と、上記仕様を満たすための機能群構造を構築するステップ(c)と、上記機能群構造の入力側及び出力側にテストベンチを付設するステップ(d)と、上記機能群構造中の機能及びテストベンチを上記アーキテクチャ中のコンポーネントに割り当てるステップ(e)とを含んでいる。

【0021】この方法により、テストベンチをHWにマッピングすることが可能になり、性能解析などにテストベンチの入出力に要する時間などのパラメータを含めることができる。

【0022】上記ステップ(a)では、上記複数のコンポーネントの中にI/Fモデルを含めておき、上記ステ

ップ(b)では、上記コンポーネント群にI/Fモデルを含めるように上記アーキテクチャを構築することにより、入出力の信号を取り扱うI/Fモデルにおける信号の遅延や消費電力を考慮したシステムの最適化が可能になる。

【0023】上記ステップ(e)では、上記入力側のテストベンチを上記アーキテクチャ中のI/Fモデルに割り当てることにより、従来、単に入力信号として扱われていたテストベンチがHWであるI/Fモデルにマッピングされるので、後の性能解析の際に遅延を考慮した処理時間の解析が可能になり、また、I/Fが動作することを考慮した消費電力の解析が可能になる。つまり、解析精度の向上を図ることができる。

【0024】上記ステップ(a)では、上記複数のコンポーネントの中にメモリを含めておき、上記ステップ

(b)では、上記コンポーネント群にメモリを含めるように上記アーキテクチャを構築し、上記ステップ(e)では、上記出力側のテストベンチを上記アーキテクチャモデル中のメモリに割り当てることもできる。

【0025】本発明の第5の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のIPを記憶装置に格納するステップ(a)と、ある仕様を満たすための機能群構造を構築するステップ(b)と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ(c)と、上記IPを機能ブロックレベルに階層展開するステップ(d)と、上記階層展開された各機能ブロックをアーキテクチャ中のコンポーネントに割り当てるステップ(e)と、上記機能ブロックから当該機能ブロックの出力を受ける他の機能ブロックまでの経路を、上記アーキテクチャ中のコンポーネント間の経路として記述する機能ブロック間テーブルを作成するステップ(f)とを含んでいる。

【0026】このように機能ブロック間パステーブルを作成することにより、動的な性能解析を精度よく行なうことができる。

【0027】本発明の第6の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のIPを記憶装置に格納するステップ(a)と、ある仕様を満たすための機能群構造を構築するステップ(b)と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するIPを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ(c)と、上記機能群中の各機能をデータベースのアーキテクチャモデル中のコンポーネントに割り当てるステップ(e)と、上記コンポーネントのバスアクセスのための優先度を設定するステップ(f)と、上記機能群構造の各機能が上記コンポーネントに割り当てられた状態で上記機能群構造の静的な性能解析を行なうステップ(g)とを含み、上記静的な性能解析においては、上記各コンポーネントのアーキテ

クチャモデル中におけるバスアクセスの優先度をトグル数と見なして上記機能群構造の消費電力を計算する方法である。

【0028】優先度をトグル数の代わりに用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【0029】

【発明の実施の形態】—IPOSデバイスの基本的な構造—

ここで配線の断面積を大きく確保しつつ、多種、多機能のデバイスを内蔵した半導体装置を実現するための1つの手段として、配線層を有する半導体配線基板例えばシリコン配線基板(Super-Sub)上に、各種デバイスを内蔵したチップIPを搭載する構成がある。その場合、各チップIP内に設けられる回路(IC)は半導体装置の設計上IP(Intellectual Property)として扱うことができ、各種IPを半導体配線基板上に貼り合わせたものと考えることができる。つまり、半導体デバイス全体は、“IP On Super-Sub”であるので、シリコン配線基板とIP群とを備えた半導体デバイス全体を“IPOSデバイス”とする。

【0030】図1(a)、(b)、(c)は、IP(チップIP)群を搭載するための配線基板となるシリコン配線基板の平面図、シリコン配線基板上に搭載されるIP群の例を示す平面図、及びシリコン配線基板の断面図である。図1(a)、(b)に示すように、シリコン配線基板10上には各種IPを搭載するための複数の領域が設けられており、各領域には、例えば、Analog-IP、Logic-IP、CPU-IP、Flashメモリ-IP、SRAM-IP、DRAM-IP、I/O-IPなどの各種IP群がチップIPとして搭載可能となっている。図1(c)に示すように、シリコン配線基板10は、シリコン基板11と、シリコン基板10上に絶縁膜(図示せず)を挟んで設けられたグラウンドプレーン12と、グラウンドプレーン12の上に層間絶縁膜を挟んで設けられた第1配線層13と、第1配線層13の上に層間絶縁膜を挟んで設けられた第2配線層14と、第2配線層14の上にパッシベーション膜を挟んで設けられたパッド15とを備えている。パッド15、各配線層13、14及びグラウンドプレーン12間は、それぞれコンタクト(図示せず)を介して所望の部位で互いに接続されている。そして、各IPは、パッド15上に貼り付けられて、各IPが配線層13、14により互いにあるいはグラウンドプレーン12に電氣的に接続される構造となっている。

【0031】シリコン配線基板10内の配線層13、14の寸法の制約は緩やかであり、数 μm 幅の配線をも設けることができるので、以下のような効果がある。経験的に、今までの半導体集積回路装置の微細化が進展した過程において、もっとも配線としての特性が良好であつ

た世代の寸法を有する配線を設けることが可能となる。また、配線の電気インピーダンスを低減することができる。

【0032】そして、シリコン配線基板上のチップIPは多くの機能を有しているが、これに対しては2つの考え方がある。1つは、できるだけ多くの機能を使用するという考え方であり、もう1つは、使用する機能を制限するという考え方である。つまり、ユーザの多様な要求に応えるためには、多種の使用方法に対応できる構成を有していることが好ましいが、反面、実際の使用に際してはいずれかの機能に限定する必要がある。つまり、この2つの相反する要求を満足させることにより、ユーザの多様な要求に応えつつ、少品種、大量生産に適したIPOSデバイスを構築することができるのである。

【0033】したがって、後に選択、制限、切り換え、設定などの処理を柔軟に行なうことを前提として、つまり、機能に関する処理の柔軟性を保持しつつ、多種の機能をハードウェア(HW)、ソフトウェア(SW)に分けて各IP(チップIP)に組み込むことが重要となる。

【0034】そこで、以下の実施形態においては、IPOSデバイスを例とした大規模なシステム(以下、「IPOSシステム」という)に適したHW/SW協調設計について説明する。ただし、本発明は、必ずしもIPOSシステムだけでなく、従来のいわゆるシステムLSIなどの大規模な集積回路システムの設計にも適用が可能である。

【0035】(実施形態)

—HW/SW協調設計全体の概略—

図2は、本発明の実施形態におけるHW/SW協調設計の概略的な手順を示すフロー図である。

【0036】ステップST1で、機能ライブラリ(機能Lib)やテストベンチを用いて機能設計を行ない、ステップST2で、IP機能Libを用いたIP機能設計を行なう。このステップST1、ST2では、半導体デバイスに搭載される各IPに要求される機能が例えばC言語により記述される。例えばJPEGの場合には、信号処理(アルゴリズム)だけが記載されているような状態である。

【0037】次に、ステップST3で、アーキテクチャモデルLibを用いてシステムのアーキテクチャ設計を行なう。システムのアーキテクチャモデルLibには、アーキテクチャモデルがシステムの設計に必要なHW、SWモデルとして存在しているので、このステップST3では、各モデルをバスに接続してシステム全体の概略的な構造を作り上げる。

【0038】次に、ステップST4で、各機能(機能ブロック)をアーキテクチャのコンポーネントに割り当てるマッピングを行なった後、ステップST5で、性能Libを用いて性能解析を行ない、その結果、不具合があ

ればステップST3のアーキテクチャ設計、又はステップST4のマッピングに戻り、性能が満足されるまでこれらのステップを繰り返すループ処理を行なう。

【0039】次に、ステップST6、ST7で、それぞれIPハードウェアモデルLibを用いたHW設計と、例えばIPソフトウェアモデルLibを用いたSW設計とを個別に行なってから、HW及びSWが詳細化されると、ステップST8で、I/Fモデルを用いたHW/SW協調検証を行なう。その後、ステップST9で、実チップを用いて実チップの機能を確認するための実チップ検証を行なう。

【0040】ー機能設計及びIP機能設計ー

図3は、機能設計を行なうステップST1と、IP機能設計を行なうステップST2との詳細な処理の内容を説明するための部分フロー図である。

【0041】ステップST1においては、以下の手順により、機能設計を行なう。機能Libには、f1、f2、f3、f4などの種々の機能(function)が格納されている。この機能f1、f2、f3、f4とは、例えば画像処理におけるMPEGKL、JPEG、Blue-toothなどの機能である。また、テストベンチLibには、tb1、tb2、tb3、tb4などの種々のテストベンチが格納されている。このテストベンチtb1、tb2、tb3、tb4は、機能設計が適正か否かを判断するための与えられるデータである。例えば、図3のステップST1中に示すように、機能f1、f2、f3を組み合わせたデザインがある場合には、入力テストベンチとして、テストベンチからtb1を取り出して、これを入力データとして与える。また、出力データとして例えば画像出力のテストベンチtb2を与える。その結果、ここで作り上げたデザインが適正かどうかがある程度検証される。

【0042】次に、ステップST2におけるIP機能設計は以下の手順で行なわれる。IP機能Libには、各機能f1、f2、f3、f4ごとに、この機能を実現するためのいくつかのIPが格納されている。例えば、機能f1に対してf1-ip1～f1-ip4があり、機能f2に対してf2-ip1～f2-ip3があり、機能f3に対してf3-ip1、f3-ip2がある。そこで、おおよその見当をつけて、デザイン中の機能f1、f2、f3に対して用いるIPを指定する。例えば、機能f1に対してはf1-ip4を、機能f2に対してはf2-ip1を、機能f3に対してはf3-ip2を用いるというようにパラメータを指定する。

【0043】次に、ステップST1で作り上げられたデザイン中の各機能f1、f2、f3に、それぞれf1-ip4、f2-ip1、f3-ip2を当てはめる。つまり、IP機能の選択を行なう。この段階では、選出された各IPはいずれもステップST1における機能f1、f2、f3を有しているので、ステップST1で検

証した結果と同じ結果が得られる。

【0044】このように、機能が互いに共通する複数のIPをLibに格納しておいて、機能ごとに適当なIPを選択することにより、IPを代えても機能の同一性が保持されるので、従来大きかった設計変更の確率をできるだけ小さくすることができる。

【0045】次に、IP機能をさらに詳細に個々の機能に分けるIP機能の階層展開を行なう。例えば、f1-ip4について階層展開をすると、f1-ip4-A（例えば逆サイン変換）を行なってからf1-ip4-B（例えば量子化）を行ない、その後f1-ip4-C（例えばハフマン符号化）を行なうという機能になっていたとする。つまり、IPであるf1-ip4を機能ブロックf1-ip4-A、f1-ip4-B、f1-ip4-Cに展開するのである。同様に、f2-ip1、f3-ip2についても階層展開を行なって、f2-ip1についてはf2-ip1-Aという機能ブロックが、f3-ip2についてはf3-ip2-A、f3-ip2-Bという機能ブロックがそれぞれ得られたとする。そして、この階層展開された各機能ブロックをデザイン中の各機能f1、f2、f3に置換する。この階層展開されたIPの中身をHW/SW分割するのが本発明の1つの特徴である。

【0046】ーアーキテクチャ設計ー

次に、図4は、ステップST3における処理の内容を説明するための部分フロー図である。同図に示すように、アーキテクチャモデルLibには、CPU、メモリ、ASIC（ロジック回路）、BUS、バスブリッジモデル、I/Fモデルなどのコンポーネント(c)についての多くの種類が格納されている。そこで、このコンポーネントごとに各種のコンポーネントパラメータをもっているのを、コンポーネントパラメータの値を与える。つまり、CPUについては動作周波数、キャッシュヒット率があり、DRAMについては動作周波数、ビット幅があり、ASICについては動作周波数、ビット幅があり、BUSについてはバスクロック、バス幅、転送レート(bps)、アービトレーション方法があり、バスブリッジモデルについてはバスブリッジ間遅延があり、I/FモデルとしてUARTを選択したときには入出力転送レート(bps)、入出力ビット幅、動作周波数、ビット幅がある。そして、各コンポーネントパラメータに具体的な値を入れる。そして、このパラメータ指定を行なってから、例えば図4に示すように、CPU1(c1)、DRAM1(c2)、ASIC1(c3)、ASIC2(c6)及びUART(c5)をBUS1、BUS2及びバスブリッジ1(c4)を介して接続した構造を有するアーキテクチャを作り上げる。

【0047】ここで、このアーキテクチャの特徴は、複数のバス(BUS1、BUS2)を備え、BUS1-BUS2間を接続するバスブリッジ1(c4)をコンポー

ネットとして設けている点である。バスは3つ以上あってもよく、このような複数のバスを設けることにより、半導体デバイスのような半導体配線基板を有する構造に適したアーキテクチャを構築することができる。複数のバスの例としては、標準バスに低速バス、高速バスなどを別途設けたものが考えられる。そして、バスブリッジモデルを設けることにより、当該バスブリッジモデルに接続される2つのバスのビット幅が互いに異なる場合などにおいてもデータの円滑な転送を図ることができる。バスブリッジモデルの例としては、16ビット幅のBUS（例えばバス1）のデータを8ビットずつ別のBUS（例えばBUS2）に送り込むバッファがある。また、アーキテクチャにおいて、I/FモデルをASICとは別に設けたことにより、入出力の信号を取り扱うI/Fモデルにおける信号の遅延や消費電力を考慮したシステムの最適化が可能になる。

【0048】—マッピング—

次に、図5は、ステップST4、ST5におけるマッピングと性能解析の概略とを説明するための部分フロー図である。

【0049】同図に示すように、ステップST4においては、ステップST2のIP機能設計によって得られたデザイン中の階層展開されたIPを、HWとSWとに割り当てる処理であるマッピングを行なう。例えば、テストベンチtb1は、HWとしてI/FモデルであるUARTに割り当てられる。また、テストベンチtb2は、HWとしてDRAM1に割り当てられる。また、階層展開されているf1-ip4中の機能ブロックf1-ip4-A(b11)はHWとしてASIC1に、機能ブロックf1-ip4-B(b12)、機能ブロックf1-ip4-C(b13)はSWとしてCPU1に、f2-ip1中の機能ブロックf2-ip1-A(b14)はSWとしてCPU1に、f2-ip2中の機能ブロックf3-ip2-A(b15)、機能ブロックf2-ip2-A(b16)はすべてHWとしてASIC2にそれぞれ割り当てられる。

【0050】ここで、このマッピングの特徴は、テストベンチをI/Fモデルにマッピングしたことである。従

来、単に入力信号として扱われていたテストベンチをHWであるI/Fモデルにマッピングすることにより、後の性能解析の際に遅延を考慮した処理時間の解析が可能になり、また、I/Fモデルが動作することを考慮した消費電力の解析が可能になる。つまり、解析精度の向上を図ることができる。

【0051】また、このように、1つのIPをHWとSWとにわけてマッピングができるようになったのは、IP機能設計において、上述のようにIPを機能ブロックのレベルまで階層展開したからである。そして、マッピングに応じて、図5の左下に示すように、各機能とデザイン中のコンポーネントとのマッピング対応表が機能番号(a1~an)とともに作成される。また、例えば機能ブロックf1-ip4-A→機能ブロックf1-ip4-Bというバスは、ASIC1→BUS1→(DRAM1→)CPU1というデータのバスとなり、機能ブロックf1-ip4-B→機能ブロックf1-ip4-Cというバスは、CPU1→(DRAM1→)CPU2というデータのバスとなること、ブロック間バステーブルとして作成され、設計装置の記憶装置(図示せず)に格納される。なお、括弧内のコンポーネントはその部分が経由される場合もあり、経由されない場合もあることを示している。

【0052】ここで、このように機能ブロック間バステーブルを作成することにより、後述するような動的な性能解析を精度よく行なうことができる。

【0053】次に、このようなマッピング結果を用いてステップST5における性能解析を行なう。このステップにおいては、静的解析と動的解析とを行なってもよいし、静的解析及び動的解析のうちいずれか一方のみを行なってもよい。

【0054】—性能解析(静的解析)—

性能解析のうち静的解析は、例えば所望のパラメータを変数としてシステム全体のコスト関数(Cost-tot)を定義して、このコスト関数(Cost-tot)を求める。そして、コスト関数(Cost-tot)が最小又はある上限値以下になるかを目標として設計を進める。このコスト関数(Cost-tot)の一例として、例えば下記式(1)

$$\begin{aligned} \text{Cost-tot} = & k1 (\text{Speed}_{\text{HW-tot}} + \text{Speed}_{\text{SW-tot}}) \\ & k2 (\text{Area}_{\text{HW-tot}} + \text{Area}_{\text{SW-tot}}) \\ & k3 (\text{Power}_{\text{HW-tot}} + \text{Power}_{\text{SW-tot}}) \end{aligned} \quad (1)$$

がある。ここで、k1、k2、k3は重み係数であり、 $\text{Speed}_{\text{HW-tot}}$ はHW全体の処理速度であり、 $\text{Speed}_{\text{SW-tot}}$ はSW全体の処理速度であり、 $\text{Area}_{\text{HW-tot}}$ はHW全体の面積であり、 $\text{Area}_{\text{SW-tot}}$ はSW全体の面積であり、 $\text{Power}_{\text{HW-tot}}$ はHW全体の消費電力であり、 $\text{Power}_{\text{SW-tot}}$ はSW全体の消費電力である。つまり、この例では、パラメータである処理速度ができるだけ大きく、面積ができるだけ小さく、消費電力ができるだけ小さくなるようにシステムを構築したいと考えていることにな

る。そこで、コスト関数(Cost-tot)は小さいほどよいとする。例えば、各種のマッピングMAP1、MAP2、MAP3を行なった結果、コスト関数(Cost-tot)が図6に示す値になったとする。このときには、もっとも値が小さいマッピングMAP1が最適であると判断する。

【0055】ここで、例えば処理速度に関する計算は、マッピング結果を用いて、下記式(2)

$$\begin{aligned} \text{Speed}_{HW-tot} (f_{b11} \sim f_{b16}, f_{bus}, R_{bus}, P_{b11} \sim P_{b16}) \\ = \text{Speed}_{HW-b11} (f_{o1}, f_{bus}, R_{bus}, R_{o1}) + \\ \dots + \text{Speed}_{HW-b16} (f_{o1}, f_{bus}, R_{bus}, P_{o1}) \end{aligned} \quad (2)$$

のように行なわれる。ここで、式(2)において、図5に示すように、マッピングしたときの機能ブロック番号をb11~b16とし、コンポーネント番号をc1~c6とする。そして、fは動作周波数であり、Rはバスの転送レートであり、Pは機能のバスに対する優先順位(優先度)(0~1の値)を表している。この優先順位

$$\begin{aligned} \text{Speed}_{HW-b11} (f_{o1}, f_{bus}, R_{bus}, P_{o1}) \\ = 1 / (f_{o1} \times f_{bus} \times R_{bus} \times P_{o1}) \end{aligned} \quad (3)$$

のように求められる。つまり、処理速度はできる限り小さい方が好ましいことから、動作周波数f、転送レートR、優先順位Pの逆数からコスト関数(Cost-tot)のうち処理速度Speedの成分を計算する。

$$\begin{aligned} \text{Speed}_{SW-tot} (St_{CPU-o1} \sim St_{CPU-o6}, f_{CPU}) \\ = \text{Speed}_{SW-b11} (St_{CPU-o1}, f_{CPU}) + \\ \dots + \text{Speed}_{SW-b16} (St_{CPU-o6}, f_{CPU}) \end{aligned} \quad (4)$$

により求められる。この計算の考え方は以下の通りである。機能ブロックをfunctionによって表すと、処理はC言語によって記述することができ、C言語で記述すればアセンブリ言語にコンパイルすることが可能となる。したがって、アセンブラのステップ数で処理速度

$$\text{Speed}_{SW-b11} (St_{CPU-o1}, f_{CPU}) = St_{CPU-o1} / f_{CPU} \quad (5)$$

のように求められる。このステップ数が小さいほど速度が大きいので、式(5)の数値がなるべく小さいことが望ましいことになる。

$$\begin{aligned} \text{Area}_{HW-tot} = \text{Area}_{HW-f1-ip4} \\ + \text{Area}_{HW-f2-ip1} + \text{Area}_{HW-f3-ip2} \end{aligned} \quad (6)$$

のように行なわれる。つまり、IP機能を選択した後でIP機能の階層展開を行なう前に使用される各IP(f1-ip4, f2-ip1, f3-ip2)の情報があるので、これからHW部分の面積を求めることができる。

【0060】次に、機能ブロックをコンパイルすると、

$$\text{Area}_{SW-tot} = (Msize_{SW-b11} + \dots + Msize_{SW-b16}) \times \text{Area}_{DRAM} \quad (7)$$

により、求めることができる。

【0061】このように、IP内が機能ブロックに分割されたことにより、機能ブロックのSWの部分からメモリサイズを計算すると、各機能ブロック間で面積を重複

$$\text{Power}_{HW-tot} = \text{Power}_{HW-b1} + \dots + \text{Power}_{HW-b16} \quad (8)$$

のように行なわれる。すると、式(8)中の例えば機能ブロックb11(f1-ip4-A)の消費電力は、下

$$\begin{aligned} \text{Power}_{HW-b11} (f_{o1}, V_{o1}) \\ = \alpha_{b11} \times f_{o1} \times (V_{o1})^2 \times P_{o1} \end{aligned} \quad (9)$$

により、表される。ただし、αは機能ブロックb11のパワー係数でIP性能Libから抽出されるもの、f_{o1}はコンポーネントc3(ASIC1)の動作周波数、V_{o1}はコンポーネントc3(ASIC1)の電圧、P_{o1}はコンポーネントc3(ASIC1)の優先順位(優先

Pは、バスに接続されるすべてのコンポーネントの優先度を合計すると1になるように設定されている。

【0056】すると、式(2)中の例えば機能ブロックb11(f1-ip4-A)の処理速度は、下記式(3)

【0057】次に、式(1)中のSWについての処理速度成分Speed_{SW-tot}は、マッピングのブロック間パステープルに基づいて、下記式(4)

を演算することができるからである。

【0058】ここで、すると、式(4)中の例えば機能ブロックb11(f1-ip4-A)の処理速度は、下記式(5)

【0059】次に、面積に関する計算は、マッピング結果を用いて、下記式(6)

アセンブラでメモリサイズがわかるので、SWのメモリの部分をすべて加算しておいて、それにエリアDRAMというデータベースを用いてDRAMの1ビット当たりの面積(単価)を計算する。それに、メモリサイズを乗じると、下記式(7)

して計算する場合があるので、これを考慮して面積の計算精度の悪化を防止するのである。

【0062】次に、HW部分の消費電力に関する計算は、マッピング結果を用いて、下記式(8)

記式(9)

度)を示している。この優先度をトグル数の代わりに用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【0063】また、SW部分の消費電力に関する計算は、下記式(10)

$$Power_{SP-tot} = Power_{SP-CPU-tot} + Power_{SP-MEM-tot} \quad (10)$$

により求められる。式(10)の第1項は、

$$Power_{SP-CPU-tot} = \alpha_{CPU} \times f_{CPU} \times Load_{CPU} \quad (11)$$

となる。ここで、 α_{CPU} はCPUのパワー係数でIP性能Libから抽出されるもの、 $Load_{CPU}$ はCPUロード(0~1)であって、「SWにマッピングされた機能ブロックのメモリサイズの合計(bit)」を「すべての

$$Power_{SP-MEM-tot} = \alpha_{DRAM} \times f_{DRAM} \times (V_{DRAM})^4 \times P_{DRAM} \quad (12)$$

となる。 α_{DRAM} はDRAMのパワー係数でIP性能Libから抽出されるもの、 f_{DRAM} はDRAMの動作周波数(Hz)、 V_{DRAM} はDRAMの電圧、 P_{DRAM} はDRAMの優先順位を示している。この優先度を用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【0065】そして、以上の計算を行なって上で、式(1)から最終的に決定されるコスト関数(Cost-tot)が最小になるか、下限値以下になったときに静的解析に関する限り、設計されたシステムが適切であると判断する。

【0066】一性能解析(動的解析)一

動的解析においては、面積Areaについては静的解析と同じであるので、処理速度Speedと消費電力Powerのみを求める。動的解析においては、設計システムに実際にデータを入力させて動作させるシミュレーションを行なう。したがって、実行時間(Time(sec))や、各機能ブロックの入出力データ、トグル率がシミュレーションから得られる。

【0067】そこで、動的解析においては、基本的には、この得られた要素を上記式(1)~(12)に代入することにより、コスト関数(Cost-tot)の値を求めることができる。その際、式(1)における(Speed_{HW-tot}+Speed_{SW-tot})は、シミュレーションで得られた時間Timeに置き換えることができ、式(9)における各コンポーネントの優先順位Pはシミュレーションで得られた各コンポーネントのトグル数で置き換えることができる。そして、式(11)におけるCPUロード $Load_{CPU}$ は、実際のシミュレーションから求まる平均ロードを用いる。

【0068】さらに、式(12)におけるDRAMの優先順位P(優先度)はシミュレーションで得られたDRAMのトグル数で置き換えることができる。

【0069】一HW/SW分割(静的分割)一

次に、ステップST4、ST5と共に行なわれるIP内のHW/SW分割について説明する。HW/SWには静的分割と動的分割とがあるが、まず、静的な分割について説明する。

【0070】図7は、静的なHW/SW分割を説明するための図である。同図に示すように、半導体デバイスなどのシステムに利用されるハードIPには、機能A、機能B、機能C、機能Dがあり、かつ、これらの機能のO

機能ブロックをSWにマッピングしたときのメモリサイズの合計(bit)」で除したものである。

【0064】一方、式(10)の第2項は、

N・OFFを設定するための設定レジスタがある。ただし、HW/SW分割における「機能」とは本実施形態における「機能ブロック」を意味している。一方、IP性能Libには、各機能A、機能B、機能C、機能DについてのSWモデルが例えばC/C++言語によって記述されて格納している。このとき、各機能をHWとSWとによって動作させた結果、1つの機能を変更、修正、削除する必要が生じる場合がある。ここで、例えば、設定レジスタ内の機能DをOFFにすることで、例えば機能Dへのクロックの供給を停止するなどの制御を行なって、機能Dが動作しないようにすることができる。これにより、無駄な電力の消費を抑制して消費電力の低減を図ることができる。また、各機能A~D間の接続を変更することもできる。

【0071】さらに、例えば1つの機能DをHWからIP性能Libに用意されたSWモデルに切り換えたり、用意しているLib内のSWモデルを修正することもできる。つまり、いずれかのLib内に各機能と等価なSWモデルを準備しておくことにより、1つのIP内をHW/SW分割することができ、ハードIPの機能が不足したときにも設計を進めることができる。また、SWの機能を修正、追加することにより、システムの変更に柔軟に対応することができる。

【0072】一動的なHW/SW分割一

次に、動的なHW/SW分割について説明する。図7に示すHW/SW分割は、HW、SWの設定を行なった後は、原則としてその後の修正は行なわない。それに対し、動的なHW/SW分割、つまりHW/SW動作の切り換えにおいては、機能(機能ブロック)が動作しながら柔軟に変更、追加、修正を受けることができる。このような動的なHW/SW分割の例として、以下、第1~第3の具体例について説明する。

【0073】1. 第1の具体例

第1の具体例では、図7に示すIPにおいて、各機能のON、OFFをSWによって制御し、かつ、停止させる場合はIP性能Libから同等のSWモデルを呼び出してこれを用いる。設定レジスタへの制御信号により、各機能を動作させながらHW/SWの切り換えを行なうことができる。

【0074】2. 第2の具体例

図8(a)、(b)は、動的なHW/SWの切り換えにおける消費電力Powerのタイムチャート図である。ま

た、図9は、このHW/SWの切り換え制御のためのプログラムの例をC言語で示す図である。図9に示すように、設定レジスタの制御により、ハードIPの機能Cを動作させた場合に、図8(a)に示すように、IPの消費電力Powerが消費電力の上限値MAXpowerを越える時があるとする。動的解析からどの機能が動作しているかが時間軸に応じてわかるので、あるレジスタに各機能の消費電力の合計を計算しながら書き込んでおいて、その値が上限値MAXpowerを越えるときを検出すればよい。そして、IPの消費電力Powerが消費電力の上限値MAXpowerを越えると、図9に示すように、設定レジスタの制御によりハードIPの機能CをOFFにして、代わりにSWモデルからfunc-Cを呼び出してこちらを動作させる。これにより、図8(b)に示すように、消費電力Powerが上限値MAXpowerを越えることがなくなり、所望の制約条件の下での作動を確保することができる。つまり、この例では、当該IPの機能A、Cが動作する期間には機能CがSWによって実現にされるが、機能A、B、Cが動作する期間には機能CがHWによって実現されるように設計することになる。

【0075】3. 第3の具体例

図10(a)、(b)は、動的なHW/SWの切り換えにおける処理速度Speedの消費電力依存性を示す図である。また、図11は、このHW/SWの切り換え制御のためのプログラムの例をC言語で示す図である。図11に示すように、CPUの設定レジスタの制御により、CPUのロード(%)がある値IDLEよりも大きいときには機能CをSWにより実行し、CPUのロード(%)がある値IDLE以下のときに機能CをHWにより実行するよう制御する。つまり、機能Cを常にHWで実行すると、図10(a)に示すように処理時間Timeが上限値MAXtimeを越える事態が発生する。そこで、レジスタにCPUのロード(%)を検知しながら書き込んでおいて、CPUのロード(%)が下限値IDLEを越えるときを検出しながら、CPUのロード(%)がある値IDLE以下のときには機能CをHWにより実行する。そして、CPUのロード(%)が下限値IDLEを越えると、設定レジスタの制御によりハードIPの機能CをOFFにして、代わりにSWモデルからfunc-Cを呼び出してSWを動作させる。このような制御を行なうことにより、図10(b)に示すように、処理時間Timeが上限値MAXtimeを越えることがなくなり、所望の制約条件の下での作動を確保することができる。

【0076】-HW/SW協調検証-

次に、ステップST8の協調検証について説明する。上述のステップST4のマッピング処理、ステップST5の性能解析処理に基づいて最終的に決定されたHW、SWの構成に従い、ステップST6、ST7で、HW設計とSW設計とを行なうと、HWモデルとSWモデルとをそれぞれ作成する。HWモデルを作成する際にはIPハ

ードウェアモデルLib(BCA, RTL又はゲートレベルで記述)を利用し、SWモデルを作成する際にはIPソフトウェアモデルLib(C言語, C++言語又はアセンブラで記述)を利用する。

【0077】次に、ステップST8では、ステップST6、ST7で作成したHWモデル及びSWモデルを利用して、HW/SW協調検証を行なうが、そのためにはHW-SW間のI/Fモデルが必要となる。そこで、以下の手順により、I/Fモデルモデルを作成する。

【0078】まず、図13に示すようなメモリマップを作成する。このメモリマップは、SWにプログラムをコンパイルしたものである。HWは、レジスタをメモリにマッピングする。

【0079】そして、各機能ブロックのアドレスを決定し、各機能ブロックのアドレステーブルをデータベース化して、メモリマップDBに登録する。そして、図14に示すように、メモリマップDBに登録されたHWのレジスタとアドレスとの対応表をとりだして、アドレスデコーダを作成する。

【0080】バスコントローラは、各HWブロックのP(優先度)にしたがって、各バス毎のアービトレーション方法ごとに用意したHDLテンプレートを生成する。すなわち、バスコントローラは、各ハードウェアのレジスタアドレスをメモリマップDBから抽出して、関数を作成し、これをドライバ関数テンプレートとしてSWモデルに格納する。

【0081】図15は、基本ハードウェアアクセス用のドライバ関数テンプレートの一例を示す図である。このようなメモリマップDB、アドレスデコーダ、関数テンプレートを利用することにより、設計されたシステムのHWによってSWを動作させたり、システムのSWによってHWを動作させることが可能になり、HW/SW協調検証を行なうことができる。

【0082】

【発明の効果】本発明により、IPの機能に着目してより最適化された半導体集積回路装置を形成するための設計が可能になった。

【図面の簡単な説明】

【図1】(a)、(b)、(c)は、IP群を搭載するための配線基板となるシリコン配線基板の平面図、シリコン配線基板上に搭載されるIP群の例を示す平面図、及びシリコン配線基板の断面図である。

【図2】本発明の実施形態におけるHW/SW協調設計の概略的な手順を示すフロー図である。

【図3】機能設計とIP機能設計との詳細な処理の内容を説明するための部分フロー図である。

【図4】システムアーキテクチャ設計における処理の内容を説明するための部分フロー図である。

【図5】マッピングと性能解析の概略とを説明するための部分フロー図である。

【図6】各種のマッピングを行なった結果のコスト関数 (Cost-tot) の値の例を示す図である。

【図7】静的なHW/SW分割を説明するための図である。

【図8】(a), (b) は、動的なHW/SWの切り換えにおける消費電力Powerのタイムチャート図である。

【図9】消費電力を考慮したHW/SWの切り換え制御のためのプログラムの例をC言語で示す図である。

【図10】(a), (b) は、動的なHW/SWの切り換えにおける処理速度Speedの消費電力依存性を示す図である。

【図11】処理速度を考慮したHW/SWの切り換え制御のためのプログラムの例をC言語で示す図である。

【図12】協調検証を説明するための図である。

【図13】SWにプログラムをコンパイルして得られたメモリマップを示す図である。

【図14】メモリマップDBに登録されたHWのレジスタとアドレスとの対応表をとりだして、アドレスデコーダを作成する処理を示す図である。

【図15】基本ハードウェアアクセス用のドライバー関数テンプレートの一例を示す図である。

【符号の説明】

- 10 シリコン配線基板
- 11 シリコン基板
- 12 グランドプレーン
- 13 第1配線層
- 14 第2配線層
- 15 パッド

フロントページの続き

(51)Int.Cl.	識別記号	F I	ターム(参考)
G 0 6 F 17/50	6 6 4	G 0 6 F 17/50	6 6 4 A
	6 6 6		6 6 6 T
			6 6 6 X
	6 6 8		6 6 8 A
H 0 1 L 21/82		H 0 1 L 21/82	B
27/04			C
21/822		27/04	U

F ターム(参考) 5B046 AA08 BA02 KA06
 5F038 CA03 DF04 DF05 DF12 DF14
 EZ08 EZ09 EZ10 EZ20
 5F064 BB02 BB09 BB12 BB13 BB14
 BB21 EE09 EE22 EE27 EE41
 EE47 HH06 HH08 HH09 HH10
 HH12